



AFRL-RI-RS-TR-2016-170

## **AUTOMATED PROGRAM ANALYSIS FOR CYBERSECURITY (APAC)**

---

FIVE DIRECTIONS, INC

*JULY 2016*

FINAL TECHNICAL REPORT

***APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED***

STINFO COPY

**AIR FORCE RESEARCH LABORATORY  
INFORMATION DIRECTORATE**

## **NOTICE AND SIGNATURE PAGE**

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09. This report is available to the general public, including foreign nationals. Copies may be obtained from the Defense Technical Information Center (DTIC) (<http://www.dtic.mil>).

AFRL-RI-RS-TR-2016-170 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE CHIEF ENGINEER:

**/ S /**

MARK K. WILLIAMS  
Work Unit Manager

**/ S /**

WARREN H. DEBANY, JR  
Technical Advisor, Information  
Exploitation and Operations Division  
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

<b>REPORT DOCUMENTATION PAGE</b>				<b>Form Approved OMB No. 0704-0188</b>	
The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. <b>PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.</b>					
1. REPORT DATE (DD-MM-YYYY) JUL 2016		2. REPORT TYPE FINAL TECHNICAL REPORT		3. DATES COVERED (From - To) NOV 2013 – DEC 2015	
4. TITLE AND SUBTITLE  AUTOMATED PROGRAM ANALYSIS FOR CYBERSECURITY (APAC)				5a. CONTRACT NUMBER FA8750-14-C-0050	
				5b. GRANT NUMBER N/A	
				5c. PROGRAM ELEMENT NUMBER 61101E	
6. AUTHOR(S)  William Arbaugh				5d. PROJECT NUMBER APAC	
				5e. TASK NUMBER SD	
				5f. WORK UNIT NUMBER IR	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Five Directions, Inc. 5560 Sterrett Place, Suite 310 Columbia, MD 21044				8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)  Air Force Research Laboratory/RIGB 525 Brooks Road Rome NY 13441-4505				10. SPONSOR/MONITOR'S ACRONYM(S) AFRL/RI	
				11. SPONSOR/MONITOR'S REPORT NUMBER AFRL-RI-RS-TR-2016-170	
12. DISTRIBUTION AVAILABILITY STATEMENT Approved for Public Release; Distribution Unlimited. This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09.					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT  The primary goal for Five Directions during the APAC program was to measure the effectiveness and efficiency of the R&D teams in detecting malware in Android applications. In order to achieve this goal, experiments were designed to test the tools being developed by the Research and Development (R&D) teams against malicious Android applications. The design of these experiments, the performance results they generated, and the lessons learned while implementing them are the main topics of this report.					
15. SUBJECT TERMS  Malware Detection, Experimentation, Performance Testing					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT  UU	18. NUMBER OF PAGES  55	19a. NAME OF RESPONSIBLE PERSON <b>MARK K. WILLIAMS</b>
a. REPORT U	b. ABSTRACT U	c. THIS PAGE U			19b. TELEPHONE NUMBER (Include area code) N/A

## Table of Contents

<b>1</b>	<b>SUMMARY .....</b>	<b>1</b>
<b>2</b>	<b>INTRODUCTION .....</b>	<b>1</b>
<b>3</b>	<b>METHODS, ASSUMPTIONS, PROCEDURES .....</b>	<b>1</b>
<b>4</b>	<b>RESULTS AND DISCUSSION.....</b>	<b>2</b>
<b>5</b>	<b>CONCLUSIONS.....</b>	<b>42</b>
	<b>APPENDIX A – Original Application Acceptance Criteria .....</b>	<b>45</b>
	<b>APPENDIX B – Updated Application Acceptance Criteria .....</b>	<b>46</b>
	<b>APPENDIX C – Data Collection Forms .....</b>	<b>47</b>
	<b>LIST OF SYMBOLS, ABBREVIATIONS, AND ACRONYMS.....</b>	<b>49</b>

## List of Figures

Figure 1 - Experiment 1B and 1C Accuracy vs. Analysis Time.....	6
Figure 2 - Experiment 2A Accuracy vs. Analysis Time .....	8
Figure 3 - Experiment 2B Accuracy vs. Analysis Time .....	9
Figure 4 - Accuracy vs. Analysis Time for Experiment 3A .....	12
Figure 5 - Analysis Times for Experiments 2A and 3A .....	13
Figure 6 - Experiment 3A Average Analysis Times per App.....	14
Figure 7 - Experiment 3A Oracle Queries by Date.....	15
Figure 8 - Accuracy vs. Analysis Time for Experiment 3B .....	17
Figure 9 - Experiment 3B Detection Accuracy per App.....	18
Figure 10 - Accuracy vs. Analysis Time for Experiment 4A .....	21
Figure 11 - Accuracy Comparison for Experiments 3A and 4A .....	22
Figure 12 - Analysis Times for Experiments 3A and 4A .....	23
Figure 13 - Experiment 4A Average Analysis Times per App.....	23
Figure 14 - Experiment 4A Oracle Queries by Date.....	24
Figure 15 – Accuracy and Analysis Time for Experiment 5A .....	26
Figure 16 - Analysis Time Breakdown for Experiment 5A.....	27
Figure 17 - Accuracy Comparison for Experiments 3A, 4A, and 5A.....	27
Figure 18 - Analysis Times for Experiments 3A, 4A, and 5A .....	28
Figure 19 - Experiment 5A App Accuracy and Analysis Time.....	29
Figure 20 - Experiment 4A Oracle Queries by Date.....	30
Figure 21 – Accuracy and Analysis Time for Experiment 5B.....	31
Figure 22 - Analysis Time Breakdown for Experiment 5B .....	33
Figure 23 - Average Analysis Time per App for Experiment 5B.....	34

Figure 24 – Accuracy and Analysis Time for Experiment 6A .....	35
Figure 25 - Analysis Time Breakdown for Experiment 6A.....	36
Figure 26 - Accuracy Comparison for Experiments 3A - 6A .....	37
Figure 27 - Analysis Times for Experiments 3A - 6A.....	37
Figure 28 - Experiment 6A Analysis Time per App.....	38
Figure 29 - Experiment 6A Oracle Queries by Date.....	38
Figure 30 – Accuracy and Analysis Time for Experiment 6B.....	40
Figure 31 - Analysis Time Breakdown for Experiment 6B.....	41
Figure 32 - Average Analysis Time per App for Experiment 6B.....	41

### List of Tables

Table 1 - Performer analysis time for Experiment 1C.....	7
Table 2 - Self-Reported vs. Actual Analysis Time (hours) .....	10
Table 3 – Experiment 3B Self-Reported vs. Available Time (hours).....	19
Table 4 - Experiment 5B Self-Reported vs. Observed Time (hours) .....	32
Table 5 - Experiment 6B Self-Reported vs. Observed Time .....	40

## 1 SUMMARY

The primary goal for Five Directions during the APAC program was to measure the effectiveness and efficiency of the R&D teams in detecting malware in Android applications. In order to achieve this goal, experiments were designed to test the tools being developed by the Research and Development (R&D) teams. The experiments pitted the research tools against malicious Android applications created by the Adversarial Challenge (AC) teams. The results of these experiments were then compared to the performance of a separate Control Team that used existing tools and techniques in order to analyze the same malicious applications. This analysis provided a method of evaluating the performances of each R&D team as well as the overall performance of the APAC program.

## 2 INTRODUCTION

The experiment timeline was divided into six engagements that took place throughout the course of the APAC program. The design of these engagements, the performance results they generated, and the lessons learned while implementing them are the main topics of this report. A high-level overview of each engagement is included which provides a narrative for how the tools being developed for APAC progressed during the program. Following this overview each engagement is analyzed in detail, including the accuracy scores and analysis times for each R&D team.

## 3 METHODS, ASSUMPTIONS, PROCEDURES

### 3.1 Engagement Workflow

Each Engagement began with the R&D Teams continuing development on their tools. During this time the AC Teams also started development on new challenge applications for the upcoming experiments. Early in each Engagement the R&D teams delivered a current snapshot of their tools to Five Directions who then securely distributed these tools to the AC teams. These snapshot deliveries allowed the AC Teams to test their applications under development against each performer's tool to see how well they performed in detecting hidden malware.

The next step in the Engagement Workflow was the delivery of all challenge applications from the AC Teams to Five Directions. Once the AC Teams delivered their applications, Five Directions then reviewed them and verified that they met the Application Acceptance Criteria (detailed in Appendix A and B). This set of criteria was introduced for Engagement 2 based on the lessons learned in Engagement 1. It was meant to ensure that all parties to the experiments

were informed about what was required for and what to expect from the challenge applications. Any application that did not meet the requirements was sent back to the AC teams in order to fix the discrepancies.

The challenge applications were also organized and separated into buckets for each upcoming experiment. The goal of this separation was to ensure that there was an even distribution of applications for each experiment based on:

- Type of attack used by the app (confidentiality, integrity, availability, benign)
- Size of the application in lines of code
- Attacks against specific R&D tools

Following the organization of the applications the Experimental phase of the Engagement began. These experiments ranged from multi-week ‘take-home’ experiments where the R&D teams analyzed the applications in their own labs on their own time, to ‘in-house’ experiments where the teams were monitored as they analyzed applications in the same room under a controlled time limit. Details for all of the experiments for each Engagement of the APAC program are discussed in the Results and Discussion section.

### 3.2 On-site and Off-site Experiments

The experiments used in evaluating the APAC tools consisted of on-site and off-site experiments. Off-site experiments provided the performers a longer duration to analyze programs. They also offered the teams more control over the number of analysts used, and greater flexibility in scheduling the analysis. One drawback to off-site experiments was that the analysis time for each challenge program was self reported, and could not be directly observed by Five Directions.

On-site experiments offered a more precise measurement of the analysis times reported by the performers. These experiments also allowed for a more reliable measurement of the number of analysts used by each team since the analysts involved could be directly observed and counted. A ceiling on the average program analysis time for each performer could be directly calculated during on-site experiments by multiplying the total time each team used during analysis by the number of analysts on the team. Dividing this total person-hour count by the number of challenge programs then provides the average analysis time per challenge.

## 4 RESULTS AND DISCUSSION

### 4.1 Initial Experiments and Results

The initial engagement of the APAC program was an exercise in understanding how to effectively evaluate the tools being developed by the R&D teams. The malicious applications that were developed for Engagement 1 were less complicated than the applications used in later

engagements. The R&D tools were not fully developed at that point, and the goal of the experiments was to exercise them against relatively simple malware.

The first experiment in Engagement 1 was Experiment 1A. During this experiment the R&D teams had one month to analyze 26 applications on their own time. This was followed by another take-home experiment three weeks later and an on-site experiment the following month. This experiment workload proved to be too frequent for the R&D teams.

Engagement 2 was designed to reduce the experiment workload for the R&D teams. It consisted of one take-home experiment and one on-site experiment. These experiments were also separated by a longer duration in order to give the R&D teams time to improve their tools.

The complexity of the malicious applications used for Engagement 2 was also increased in order to evaluate how the R&D tools performed in detecting more advanced malware. In Experiment 2A, the analyst teams found numerous unintended malware detections in these more advanced applications. These issues ranged from simple bugs to actual security issues. In order to keep the analysts searching for the intended malware, the Oracle question and answer system was implemented for Experiment 2B (further details of this system are discussed in the Lessons Learned section below).

## 4.2 Continued Tool Improvement

Beginning with Engagement 3, the progress that the R&D teams had made in improving their tools became more evident. The Oracle system was keeping the analysts on track, and their analysis results provided improved measurements of their ability to find increasingly advanced malware.

Experiment 3A produced a number of important results. MIT and Utah significantly improved their performance and achieved a detection accuracy of 90%. Unlike in previous experiments, in this case the highest detection accuracies also had the longest analysis times. Most teams spent significantly more time analyzing the apps than in previous experiments. The Control Team, however, did not make use of the Oracle system, and subsequently only made one correct detection, resulting in a detection accuracy of 10%.

Experiment 3B proved to be a difficult challenge for the analysts. Five of the teams were unable to make more than one correct detection during the entire experiment. The majority of these single detections were for the easier to classify benign Vermilion application. MIT had the highest detection accuracy for Experiment 3B and correctly analyzed three out of the four applications. As was seen in Experiment 3A, the highest detection accuracy in Experiment 3B also required the longest analysis time.

Engagement 4 was modified in order to have only one experiment. Experiment 4A continued the trend of increasing accuracy for all R&D teams during the take-home experiments. Unlike Engagement 3, in this case the two most accurate teams (MIT and UCSB) also had the shortest

analysis times. The Control Team was only able to analyze seven of the fourteen malicious applications, and only correctly detected one application.

Engagement 5 also continued the trend of increasing accuracy for the tools being developed for the APAC program. The number of R&D teams that were able to correctly analyze all of the malicious applications increased to three during Experiment 5A, whereas only two teams were able to achieve this in Experiment 4A. These three teams (MIT, UCSB, and Utah) were differentiated by their widely varying analysis times.

During Experiment 5B, four of the five teams were able to correctly analyze each application (MIT, Stanford, UCSB, and UW). The standout performance during Experiment 5B was Stanford. Stanford was able to correctly analyze each application, and they only required two hours and forty minutes to complete the entire experiment. The remaining teams required the entire experiment duration for their analysis.

Engagement 6 was the final engagement of the APAC program. The results of Experiment 6A showed that the improved performances of the tools being developed for the APAC program continued. The number of R&D teams that were able to correctly analyze all of the malicious applications continued to be three during Experiment 6A. These three teams (MIT, UCSB, and Utah) were differentiated by their widely varying analysis times.

Finally, in Experiment 6B MIT and Stanford found the malware in all applications. Stanford had the shortest average analysis time and Utah had the longest. UCSB found 75% of the malware, Utah found 50% of the malware, and UW correctly analyzed one application.

### 4.3 Keeping Up with Android Advancements

There was an extended break between Engagements 3 and 4 that provided the opportunity to update the Android platform version that was being targeted by APAC. All previous experiments targeted Android 4.0.3 Ice Cream Sandwich (API Level 15), which was released in December 2011. Engagement 4 targeted Android 4.4 KitKat (API level 19), which was the latest version at the time. This upgrade offered the AC Team new Android features to exploit and also focused the R&D tools on the current state of Android development.

In conjunction with the updated Android target version, the Application Acceptance Criteria was also updated to include the latest versions of the standard Android build tools (see Appendix B – Updated Application Acceptance Criteria). This included the latest version of the Eclipse IDE as well as the latest version of the ADT development plugin.

Another update was the permitted use of Google Play Services APIs by the challenge applications developed by the AC team. These APIs are an addition to the standard Android APIs, and offer access to functionality such as Google Maps and Google+. Since Google does not release source code for these functions, some of the R&D teams needed to model these new APIs. As in previous engagements, the list of APIs that were used by the challenge applications

were delivered to the R&D teams prior to the experiments in order to allow ample time for this modeling effort.

#### 4.4 Detailed Engagement Review

The following sections analyze the performance results for each Engagement of the APAC program. This analysis contains the following topics for each experiment:

- Accuracy and analysis times for each R&D team
- Comparisons of performance results with previous experiments
- Analysis difficulty for each malicious application
- Details of the oracle queries processed for each experiment

#### 4.5 Engagement 1

Engagement 1 consisted of three experiments: two “take-home” experiments and one on-site experiment during an APAC PI meeting. Experiment 1A was a five week “take-home” test that began on November 30<sup>th</sup>, 2012 and concluded on January 7<sup>th</sup>, 2013. In this experiment the performers were given twenty-six applications to analyze in their own labs. The performers were asked to submit a response document in which they would report for each application where they found malware, what triggered the detection, the amount of time spent analyzing each app, and how their tool helped in analyzing the applications.

As we reviewed the response documents after the experiment was completed, we noticed that while some of the teams reported the relevance of their tools, many either failed to report this or provided a vague response. This lack of response prevented us from being able to quantify how much of the analyses was completed using tools, and how much was done manually. Since this program is focused on how the tools can impact the effectiveness and efficiency of the analyst in finding malware, it is important that we can make the distinction between tool use and purely manual analysis. To prevent this from happening in the future, the per-app response documents were modified for the next experiment to emphasize the importance of reporting tool relevance and accurate time keeping.

Another lesson learned from Experiment 1A was that BBN’s anti-diffing technology caused some performers to complain that it was too close to obfuscation, which was not allowed in the rules of engagement. BBN’s goal with using this technology was to make their open-source apps more difficult to compare to the projects on which they were based, since not doing this would presumably result in easy detection of hidden malware. The anti-diffing consisted mainly of using a thesaurus to rename variable and function definitions. However, using this technology resulted in producing names that were unintelligible to a human reading the code. Therefore the use of anti-diffing was discontinued and ‘clean’ versions of these apps were released during the experiment.

#### 4.5.1 Experiment 1B

Experiment 1B was a two week “take-home” test that began on January 28<sup>th</sup>, 2013 and concluded on February 11<sup>th</sup>, 2013. In this experiment the performers were given sixteen applications to analyze in their own labs. As discussed in the preceding section, the response documents for this experiment were modified in order to place an emphasis on reporting the relevance of performers’ tools in analyzing the applications. In this experiment the performers were more compliant<sup>1</sup> in reporting this information and we were able to determine when the tools were relevant and when they were not.

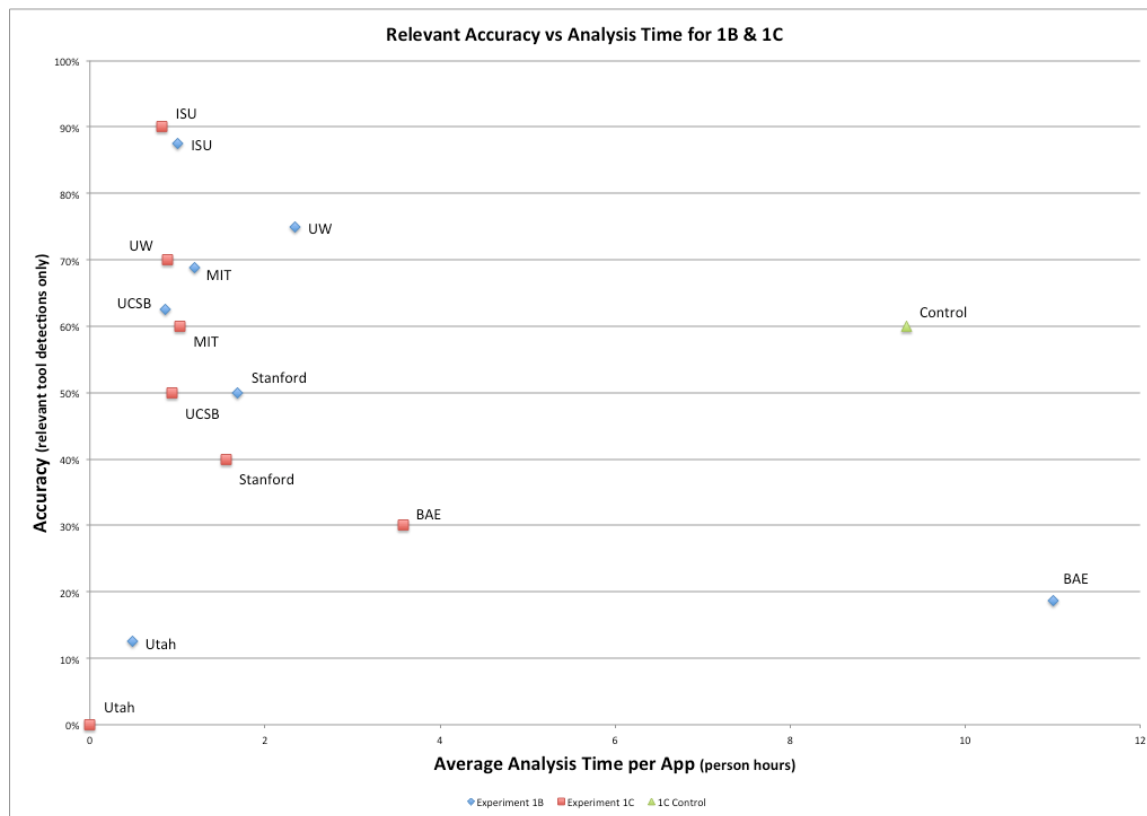


Figure 1 - Experiment 1B and 1C Accuracy vs. Analysis Time

An issue that arose during this experiment was that some teams were relying on building the test applications on the command line. The AC teams were working under the assumption that Eclipse was the only build environment, so they did not verify that their apps would build

<sup>1</sup> We still believe, however, that an over-reporting bias exists here.

correctly on the command line. The App Acceptance Criteria for Engagement 2 has been updated to include both Eclipse and command line building using Ant as requirements for all apps submitted by the AC teams.

#### 4.5.2 Experiment 1C

Experiment 1C took place over the course of 2 days during the February Utah PI meeting on February 20<sup>th</sup> and 21<sup>st</sup>, 2013. Performers were given 5 hours per day to analyze 10 applications each in a room where the count of team member participation and total analysis time could be more tightly monitored. This allowed us to obtain a per-app analysis time that is independent of the times that were self-reported by the teams. Based on the number of team members involved in analyzing apps for each team, and the time at which they submitted their results, the following table shows the average per-app analysis time for each team during Experiment 1C, as well as the self-reported times they provided.

**Table 1 - Performer analysis time for Experiment 1C**

<b>Team</b>	<b>Average Analysis Time (person hours)</b>	<b>Self-Reported Time Average for 1C</b>
<b>BAE</b>	7.0	3.282
<b>ISU</b>	3.5	0.822
<b>MIT</b>	2.8	1.02
<b>Stanford</b>	3.1	2.04
<b>UCSB</b>	3.0	1.215
<b>Utah</b>	2.75	1.595
<b>UW</b>	3.33	0.847

One item to note with these times is that they include the time spent writing the result documents. The teams did not necessarily account for this report writing time in the self-reported times, although this wouldn't fully account for the discrepancy. Also, in BAE's case, there were some apps that they started to analyze, but did not complete. Time spent on these apps was not included since it would artificially lower their per-app analysis time.

Experiment 1C was the first experiment to include an outside Control Team that analyzed the same applications using existing tools and techniques. For Experiment 1C, the Control Team's analysis had a 60% detection rate and a mean analysis time of 9.33 person hours per application. In comparison to the APAC performers the Control Team had a higher detection rate than BAE, Stanford, USCB, and Utah when the relevance of the performers' tools is included. The Control Team's per-app analysis time was significantly longer than all other teams except BAE.

It is important to note that all of the Control Team analysts were not necessarily experts in analyzing malicious applications. Also, the APAC performers had two previous APAC Experiments, 1A and 1B, in which they gained familiarity with the process.

## 4.6 Engagement 2

Engagement 2 consisted of two experiments: one “take-home” experiment and one on-site experiment during an APAC PI meeting. Experiment 2A was a two week “take-home” test that began on June 24<sup>th</sup>, 2013 and concluded on July 8<sup>th</sup>, 2013. The performers were given sixteen applications to analyze for this experiment. The AC teams stepped up the difficulty of their challenge applications as can be seen in the lower accuracy and longer analysis times during this experiment. All of the R&D teams out-performed the Control Team with respect to detection accuracy, and all of the R&D teams besides BAE had lower analysis times as well.

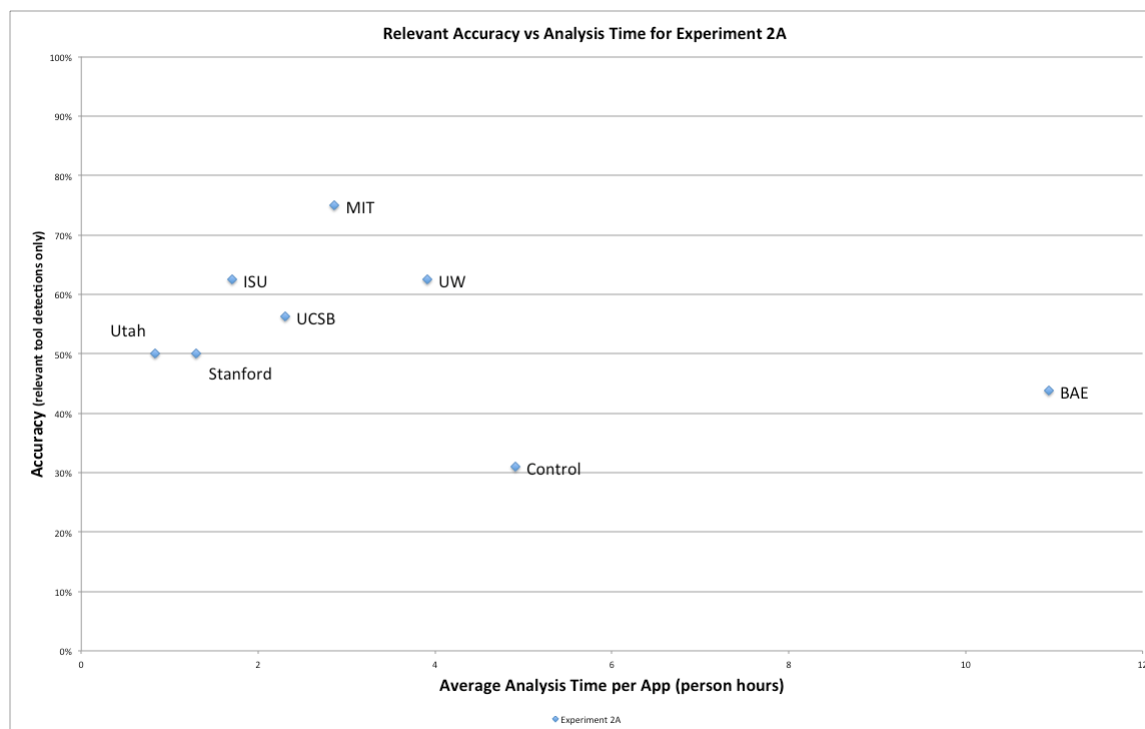


Figure 2 - Experiment 2A Accuracy vs. Analysis Time

The Control Team’s accuracy for this Experiment was quite low with only 5 correct detections out of the 16 applications. This poor performance was partially due to the Control Team ending their investigations after finding issues that could be considered bugs or poor coding practices. This caused the team to miss the larger and more effective malware placed in the applications by the AC teams. Since the other R&D teams also had issues with this type of detection short-circuiting, we introduced the Oracle System in Experiment 2B.

#### 4.6.1 Experiment 2B

Experiment 2B was held during the July Combined PI meeting on July 24<sup>th</sup>. This was an ‘in-house’ experiment where representatives from each R&D team, as well as members of the control team, analyzed eight applications during a six-hour, one-day experiment.

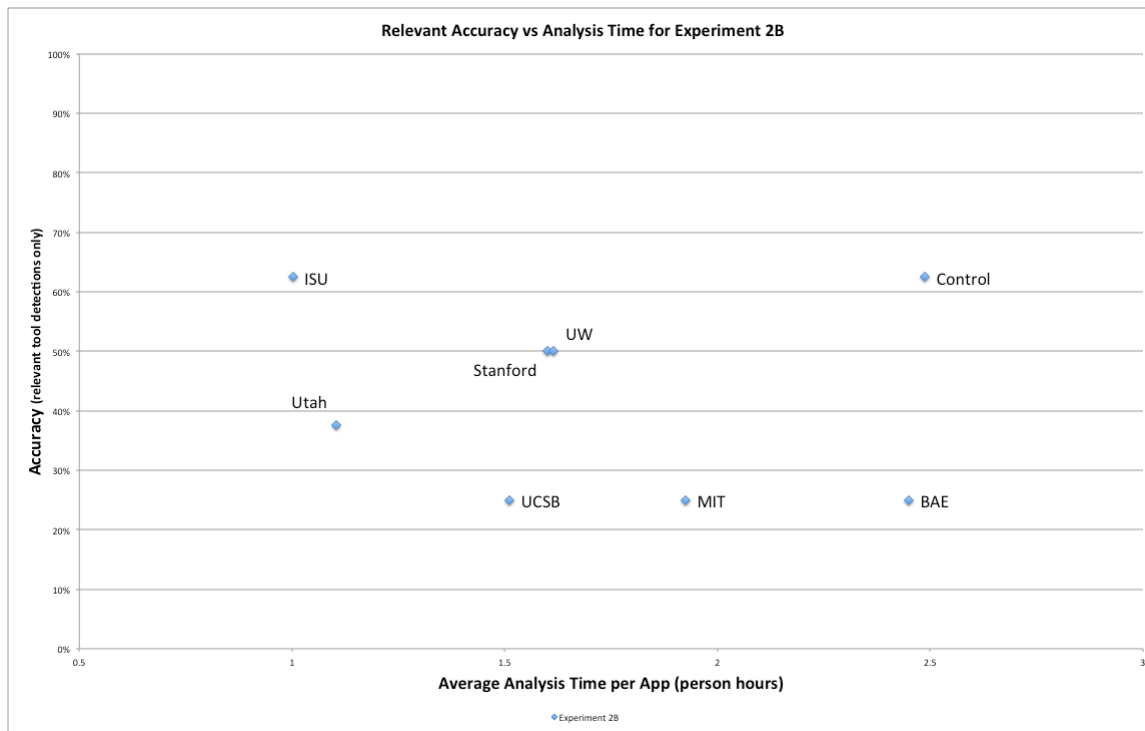


Figure 3 - Experiment 2B Accuracy vs. Analysis Time

#### 4.6.2 Analysis Times

Conducting the experiment during a single day, with all participants analyzing applications in the same room at the same time, provided better control over measuring the analysis times for each team. Table 2 lists the number of members who participated from each team, their self-reported average analysis time, and the actual analysis time available for them to complete the experiment. The available amount of time was calculated by multiplying the entire duration of the experiment (5.91 hours) by the number of participants for each team. This number was then divided by the number of applications used in the experiment. All but one team used the entire duration.

**Table 2 - Self-Reported vs. Actual Analysis Time (hours)**

<b>Team</b>	<b># of members</b>	<b>Self-Reported</b>	<b>Available Time</b>
<b>BAE</b>	3	2.45	2.21
<b>ISU</b>	3	1.00	2.21
<b>MIT</b>	3	1.92	2.21
<b>Stanford</b>	3	1.6	2.12
<b>UCSB</b>	2	1.51	1.47
<b>Utah</b>	2	1.10	1.47
<b>UW</b>	3	1.64	2.21
<b>Control</b>	3	2.48	2.21

As can be seen in Table 2, a few of the self-reported average analysis times were greater than the calculated available amount of time. Based on the reported numbers, it appears that this is due to automation being run on one application while the team members were analyzing another application. Another item to note with the calculated available times is that they include the time spent writing the result documents. The teams did not necessarily account for this report writing time in the self-reported times, which explains a portion of the discrepancy.

In Experiment 2B the Control Team's performance matched that of the highest R&D team with respect to accuracy. Though their analysis times were the slowest of all teams, the gap between their performance and the rest of the R&D teams was considerably smaller than in earlier experiments.

#### **4.7 Engagement 3**

Engagement 3 consisted of two experiments, a take-home experiment in early March 2014, and an onsite experiment at the APAC PI meeting in Pittsburgh in April 2014. The number of experiments for Engagement 3, as well as their duration, was updated based on lessons learned during the two previous APAC engagements. The design of this engagement sought to balance the impact that the experiments would have on the R&D teams' research efforts with the need to measure the performance of their tools. The spacing between the two experiments was also increased in order to allow the R&D teams sufficient time to adapt their tools based on the lessons they learned from the first experiment.

The following sections describe the Engagement 3 experiments in detail and discuss the performance results that were gathered.

##### **4.7.1 Experiment 3A**

Experiment 3A was a one work week take-home experiment that started on March 3<sup>rd</sup>, 2014 and ended on March 7<sup>th</sup>, 2014. It was mentioned during the recent site visits that these take-home experiments consume the R&D team's attention throughout the duration of the experiment. This

leads the teams to pause their research and development during this period. The previous take-home experiment (Experiment 2A) had a duration of two full weeks. The duration of Experiment 3A was therefore shortened to one work week in order to reduce the analysis burden on the R&D teams.

The number of applications to analyze during this experiment was also reduced in order to match the reduced duration. There were ten applications in this experiment whereas the previous take-home experiment contained sixteen.

#### 4.7.2 Accuracy and Analysis Times

When compared to previous take-home experiments, Experiment 3A produced a number of important results. Here are the key points:

- MIT and Utah significantly improved their performance and tied with the highest detection accuracy (90%)
- Unlike in previous experiments, in this case the highest detection accuracies also had the longest analysis times
- ISU, Stanford, and UCSB had detection accuracies in the 60%-70% range
- UW scored the lowest of the R&D teams with a detection accuracy of 30% (they only used the Oracle system a few times)
- Most teams spent significantly more time analyzing the apps than in previous experiments
- The Control Team, however, did not make use of the Oracle system, and subsequently only made one correct detection, resulting in a detection accuracy of 10%

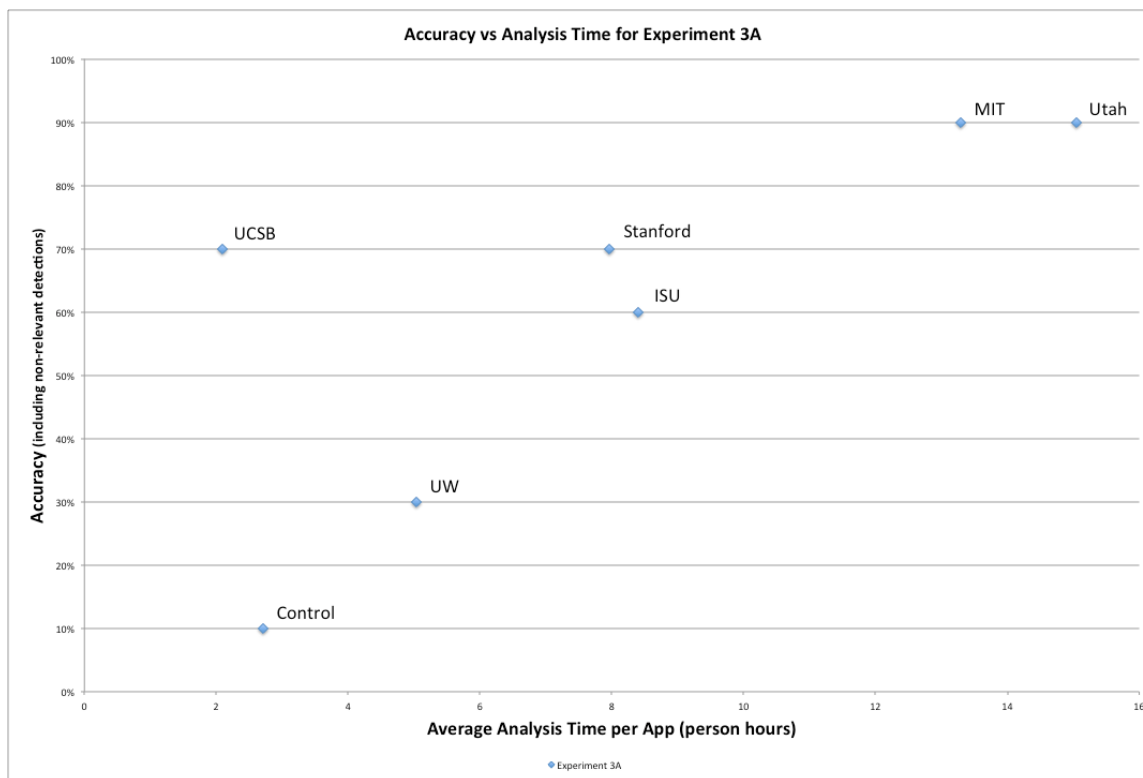


Figure 4 - Accuracy vs. Analysis Time for Experiment 3A

When compared to the previous take-home experiment (Experiment 2A), the increase in analysis times is significant. This is a direct result of the Oracle system keeping the analysts from short-circuiting on unintentional malware. The analysis times for Experiment 3A provide a more realistic measurement of the time required to find the more difficult malware in the applications. In Experiment 2A, the Oracle system was not in use, and the teams halted their analysis on the first suspicious functionality that they encountered. Frequently the functionality they encountered was easier to find than the intended malware placed in the applications by the AC team.

Figure 5 shows the average analysis times per team for Experiment 2A and 3A.

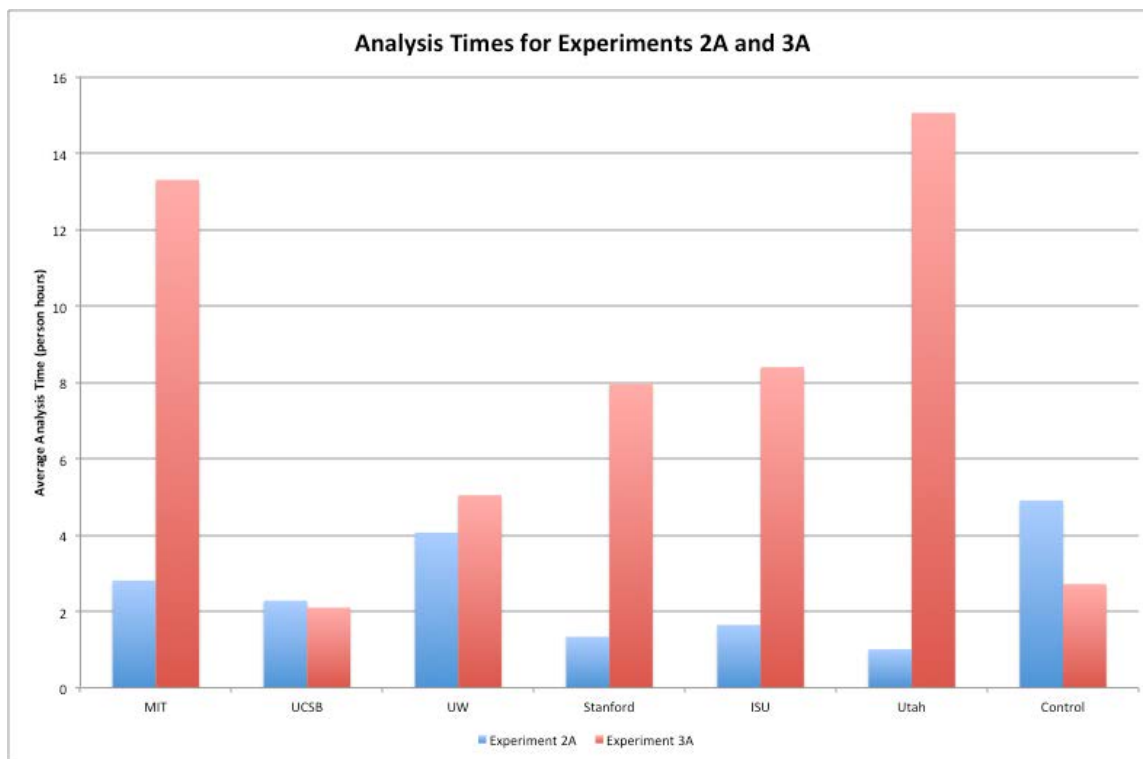


Figure 5 - Analysis Times for Experiments 2A and 3A

The average analysis time per application (Figure 6) also demonstrates that deciding when to cease analysis is not always a clear-cut decision. The two applications with the longest analysis times in Experiment 3A were Entomologist and Meetloaf. Entomologist was the only application in this experiment that did not have any correct detections. Not knowing if this application was benign or not, the teams continued to search for any hidden malware until the end of the experiment's time limit. Similarly, Meetloaf was a benign application that had the second longest analysis time. In this case, since there was no hidden malware contained in the application, the teams had no indication of when to stop looking.

The implication of this difficulty in determining when to stop analysis is that in a real-world application store, where benign applications would likely be vastly more prevalent, average analysis times could be much higher than in the experiments. Finding malicious functionality is a clear indication to the analyst that they can stop analysis. However, if the majority of the applications being analyzed are benign, then a clear indication of when to stop analysis is not present. Figure 6 shows the average analysis times for each application in Experiment 3A.

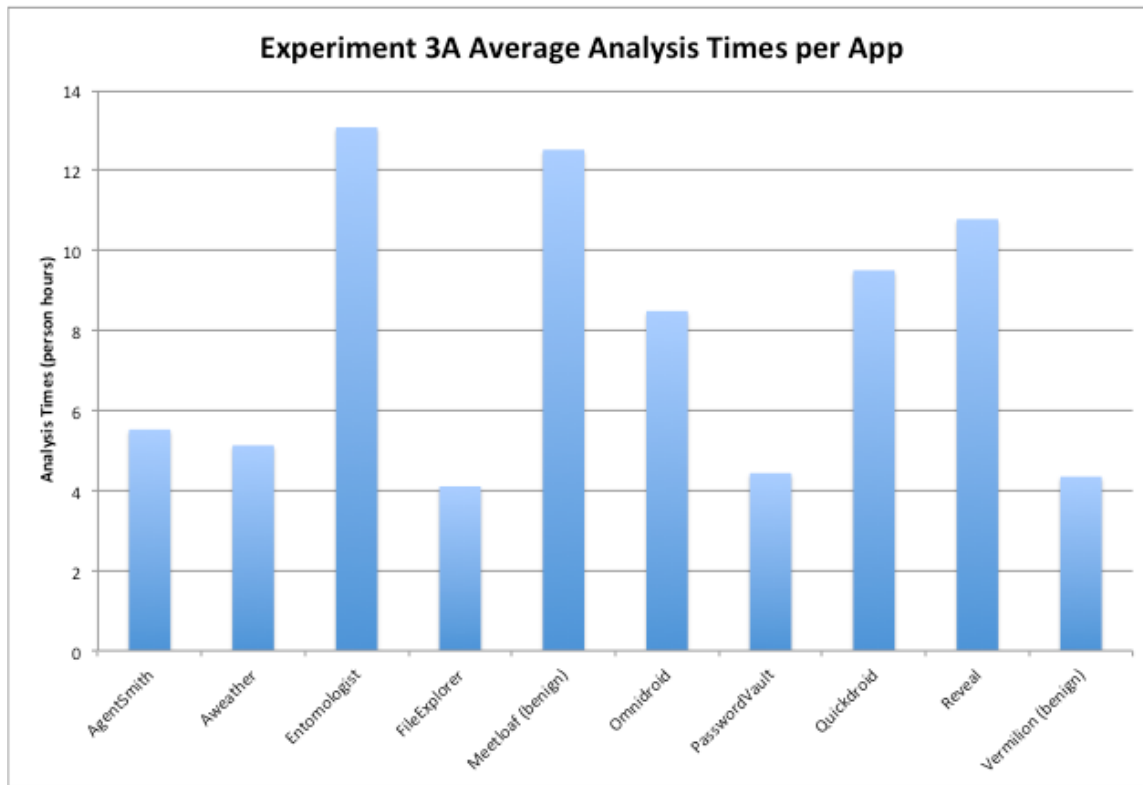


Figure 6 - Experiment 3A Average Analysis Times per App

#### 4.7.3 The Take-Home Oracle

The Oracle system was successful in keeping the R&D teams searching for the intended malware during Experiment 3A. There were over one hundred distinct Oracle email queries during the experiment. Figure 7 shows the number of Oracle queries from each team for each day of the experiment.

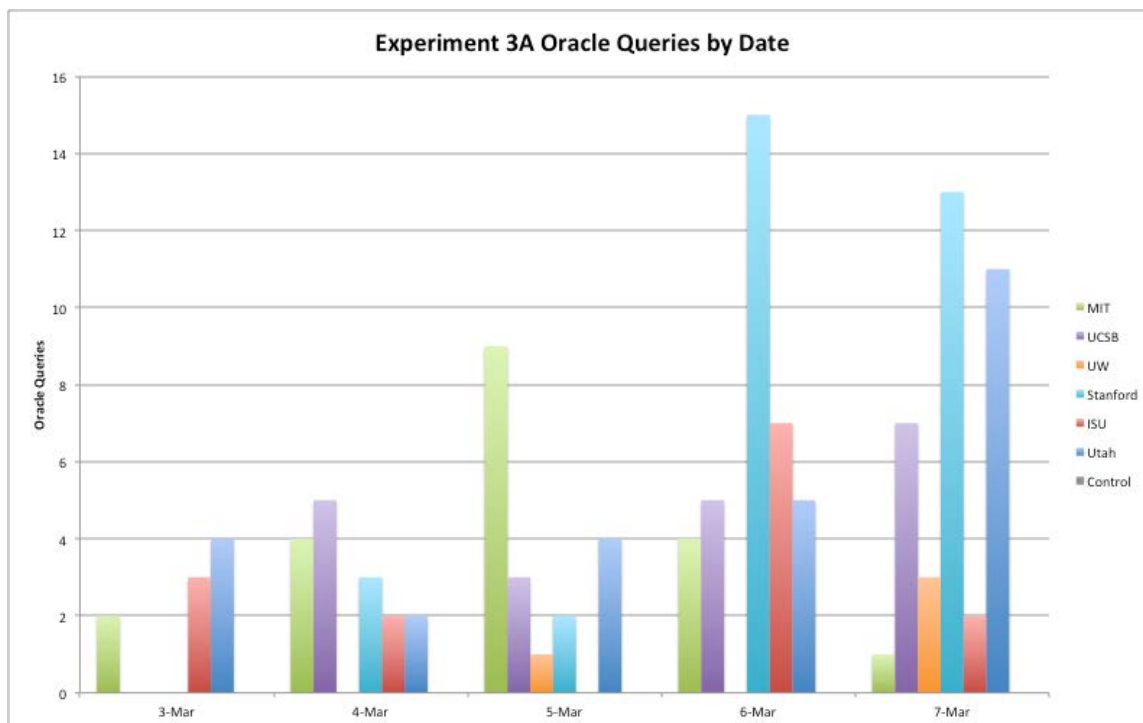


Figure 7 - Experiment 3A Oracle Queries by Date

#### 4.7.4 Unintended Malware Scoring

During this experiment, the R&D teams reported numerous instances of malicious functionality in the applications that were not the intended malware. Once the experiment was completed, an analysis of this unintended malware began. The goal of this analysis was to determine if the detection accuracy scores for the teams should be adjusted based on any unintended malware they reported. To accomplish this, all Oracle emails and result documents from Experiment 3A were reviewed and a list of all reported malicious functionality for each application was created. This list was then summarized by combining issues that were reported by multiple teams.

Once completed, this unintended malware list was sent to DARPA in order to obtain guidance on which issues might be important enough to alter the detection accuracy scores. Based on this process, the detection accuracy scores were adjusted for the following three applications:

- Meetloaf: Although this app was originally intended to be benign, correct detection scores were added for any team that reported that the application was vulnerable to malicious web input from other applications on the device. All other detections were not scored as correct, including any reports stating that the application was benign.
- PasswordVault: In addition to the intended malicious functionality, correct detection scores were added for any team that reported that passwords were being copied to the clipboard. This also included reports that this application colluded with AgentSmith to

leak passwords from the clipboard to the Internet. Teams that reported one or more of these malicious features were scored as correct.

- Vermillion: Although this app was originally intended to be benign, correct detection scores were added for any team that reported that images and metadata were being sent over the network without encryption, that the application saves images and metadata to the external SD card, or that the application leaked information to the system log. Teams that reported one or more of these three malicious features were scored as correct. All other teams were scored as incorrect, including those who reported the application as benign.

These changes did not affect the experiment results significantly. These adjustments to the scoring improved the detection accuracy of Stanford, UCSB, and the Control Team by 10%, while the detection accuracy for ISU and UW was decreased by 10%.

#### 4.7.5 Control Team Performance

The Control Team's performance during Experiment 3A was disappointing. The Control Team did not send any Oracle inquiries during the entire experiment. The team also did not deliver their result documents until the day after the experiment concluded. Without the guidance of the Oracle, the Control Team originally scored zero correct detections. This score was later increased to one correct detection due to the unintended malware scoring adjustments.

When asked to explain their performance, the Control Team responded with the following explanations:

- They struggled with the increased difficulty of the applications.
- They lost experience having only two analysts return from the previous experiments. Losses were due to departures from their school.
- They did not have enough time to fully analyze the apps since the experiment came at the end of their academic quarter when the students were completing projects and studying for finals.
- Finally, they didn't have any Oracle queries because they had difficulty finding anything that they thought was malicious.

It is important that the Control Team fixes their experience losses and time conflicts for future APAC experiments. Without a reliable performance from the Control Team it will be difficult to compare their results to those of the R&D teams.

#### 4.7.6 Experiment 3B

Experiment 3B took place on April 24<sup>th</sup>, 2014 at the APAC PI Meeting in Pittsburgh. The duration of this experiment was five hours and each team was allowed to use three analysts. Based on lessons learned from previous experiments, the number of applications that were selected for this experiment was reduced to four. This reduction was prompted by the analysis

time results from Experiment 3A. Five of the seven teams in that experiment had average analysis times of five hours per application or greater, with two of the teams having averages of over thirteen hours per app. With three analysts per team, and a five-hour duration for the experiment, there were a total of fifteen person hours available to each team for analysis.

The Oracle system was in effect for this experiment, and there were numerous unintended malware detections reported by the analysts. The Oracle kept the teams looking for the intended malware by making sure they did not stop their analysis prematurely.

#### 4.7.7 Accuracy and Analysis Times

Experiment 3B proved to be a difficult challenge for the analysts. Five of the teams were unable to make more than one correct detection during the entire experiment. The majority of these single detections were for the easier to classify benign Vermilion application. MIT had the highest detection accuracy for Experiment 3B and correctly analyzed three out of the four applications. As was seen in Experiment 3A, the highest detection accuracy in Experiment 3B also required the longest analysis time.

The team with the next highest detection accuracy was UW, which made two correct detections. UW was not only more accurate than five other teams, they also had a lower average analysis time than four of the teams, including MIT.

The remaining teams (Control, ISU, Stanford, UCSB, and Utah) were only able to make one correct detection. Figure 8 shows the detection accuracy and average analysis time

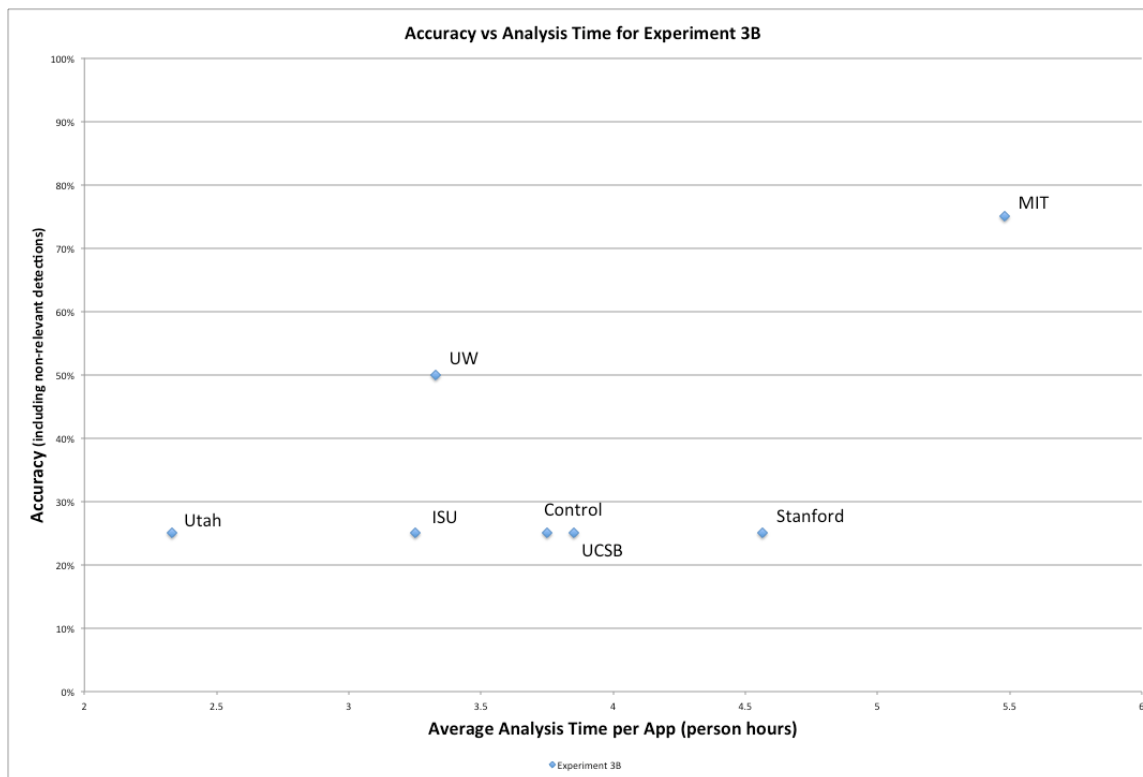


Figure 8 - Accuracy vs. Analysis Time for Experiment 3B

Figure 9 displays the average detection accuracy for each application in Experiment 3B. The obvious outlier with the highest accuracy was the benign application Vermilion. This application was an update to the previously used Vermilion base application from Experiment 3A. Vermilion had a detection accuracy of 86%, with every team except for the Control Team correctly classifying it. Benign applications appear to be easier to correctly classify for the analysts since the Oracle directs them to ignore any unintended malware they discover in them. If the Oracle rejects all of their detections, then it is relatively safe for the teams to conclude that the application is benign.

The remaining applications, each of which had hidden malicious functionality, had detection accuracies that were lower than 30%. It appears that the difficulty level of the malicious applications has reached a point where it has become unlikely that the teams will be able to successfully analyze the software during the onsite experiments.

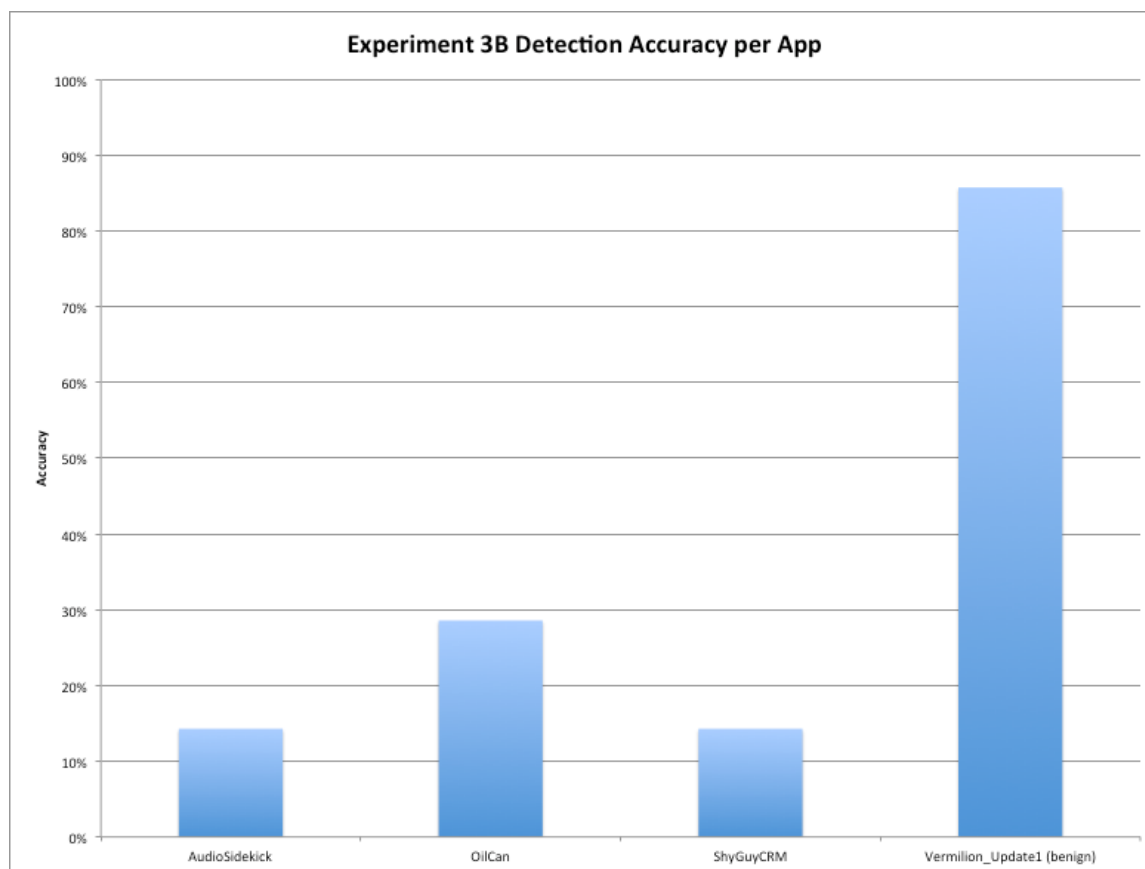


Figure 9 - Experiment 3B Detection Accuracy per App

Unlike previous in-house experiments, where some teams finished analysis early, in this experiment every team made full use of the available time for analysis. Table 3 lists the number of members who participated from each team, their self-reported average analysis time, and the actual analysis time available for them to complete the experiment. The available amount of time was calculated by multiplying the entire duration of the experiment (5.13 hours) by the number of participants for each team. This number was then divided by the number of applications used in the experiment. There were network connectivity issues during the beginning of the experiment. These issues were fixed, and all teams were able to continue analyzing the applications.

**Table 3 – Experiment 3B Self-Reported vs. Available Time (hours)**

<b>Team</b>	<b># of members</b>	<b>Self-Reported</b>	<b>Available Time</b>
<b>ISU</b>	3	3.25	3.84
<b>MIT</b>	3	5.48*	3.84
<b>Stanford</b>	3	4.56	3.84
<b>UCSB</b>	3	3.85*	3.84
<b>Utah</b>	3	2.33	3.84
<b>UW</b>	3	3.33	3.84
<b>Control</b>	3	3.75	3.84

\*includes pre-computation time

#### **4.7.8 Pre-processing Support**

Three of the teams (MIT, UCSB, and Utah) requested that they be allowed to pre-process the applications used in Experiment 3B. The automated analysis tools from these teams can sometimes take hours to complete. The goal of this pre-processing was to allow these teams to use their time during the experiment to analyze the results of their tools, instead of spending the majority of their time waiting for the tools to complete. Of the three teams that were granted pre-processing, only MIT and UCSB made use of it.

In order to support pre-processing, Five Directions hosted a virtual machine server that was capable of running pre-built virtual machines that contained each team's analysis tools and their supporting environment. In order to accommodate the large memory requirements of the tools (64GB for MIT, 16GB for UCSB) this server was set-up with 128GB of RAM.

Each team was given remote command line access to their virtual machine, and the Experiment 3B applications were made available to them starting on Sunday March 20<sup>th</sup>. In order to prevent any unfair advantage that this early access might provide, the teams were instructed to only start their automated analysis, and to not manually view the application's source code. In addition to

this, since the UCSB tool does not require source code for the automated portion of their analysis, only the pre-compiled Android .apk files for the Experiment 3B applications were provided to them.

#### 4.7.9 Unintended Malware Scoring

As with Experiment 3A, there were numerous unintended malware detections reported to the Oracle during this experiment. Once again, all Oracle emails and result documents from the experiment were reviewed, and a list of all reported malicious functionality for each application was created and sent to DARPA. Unlike the previous experiment, however, no changes to the final scoring were made based on these findings.

#### 4.8 Engagement 4

Prior to this engagement the decision was made to no longer perform on-site experiments. To accommodate this, Engagement 4 consisted of one take-home experiment that was lengthened to two full weeks (including weekends) in duration. Experiment 4A began on September 29th, 2014 and ended on October 13<sup>th</sup>, 2014. This experiment contained fourteen malicious applications to be investigated by the analyst teams. These teams consisted of six APAC R&D teams using their custom developed analysis tools, and one Control Team that used existing tools and techniques to analyze the applications. As in previous experiments, the Oracle question and answer system was in effect for Experiment 4A.

The length of this take-home experiment, and the number of applications used, was influenced by feedback from Engagement 3. The take-home experiment for Engagement 3 had a duration of one work-week. Following that experiment, it was requested by some teams that weekends be made available for the experiments in order to provide greater scheduling flexibility when analyzing applications. The analyst teams also noted that the number of applications used during Experiment 3A placed a significant time burden on them based on the duration of the experiment. In order to address these issues Experiment 4A included two weekends and a reduced application-per-day ratio (14 apps over 15 days vs. 10 apps over 5 days).

Excluding the Control Team, all of the R&D teams were able to analyze the applications during this period, with MIT actually completing the experiment after only one week.

##### 4.8.1 Experiment 4A Results

When compared to previous take-home experiments, Experiment 4A produced a number of important results. Here are the key points:

- Accuracy increased for all R&D teams
- Analysis times increased for every team except for MIT who improved their times
- The two most accurate teams (MIT and UCSB) also had the shortest analysis times
- All apps were known to be malicious
  - This led to increased analysis times since teams knew to keep looking

- Utah spent over 100 hours analyzing two applications
- The Control Team was only able to analyze seven of the fourteen malicious applications, and only correctly detected one application.

Figure 10 displays the average accuracy and analysis time results for each performer in Experiment 4A. MIT and UCSB both correctly analyzed each malicious application. Their analysis times were also the best of all of the R&D teams. This is an inversion from the previous experiments where the highest accuracy scores also had the longest analysis times. This may be an indication that these tools are beginning to distance themselves from the other R&D tools.

ISU and Utah both found the malware in all but one of the challenge applications. There was, however, a large discrepancy in their average analysis times. This discrepancy was caused by Utah spending an inordinate amount of time analyzing two specific applications. For the Memotis application, Utah spent a total of 152 hours attempting to locate the hidden malware. They also spent 114 hours analyzing Vermilion-Update3. The cause for these long analysis times is the fact that all applications in Experiment 4A were known to be malicious. Since Utah knew that there was malice in these applications, they simply kept looking for it until they either found it or ran out of time. Without these two anomalies, Utah's average analysis time would be closer to that of ISU and UW.

The Control Team had one of the shortest average analysis times, but they also had the worst accuracy score and only correctly analyzed one application.

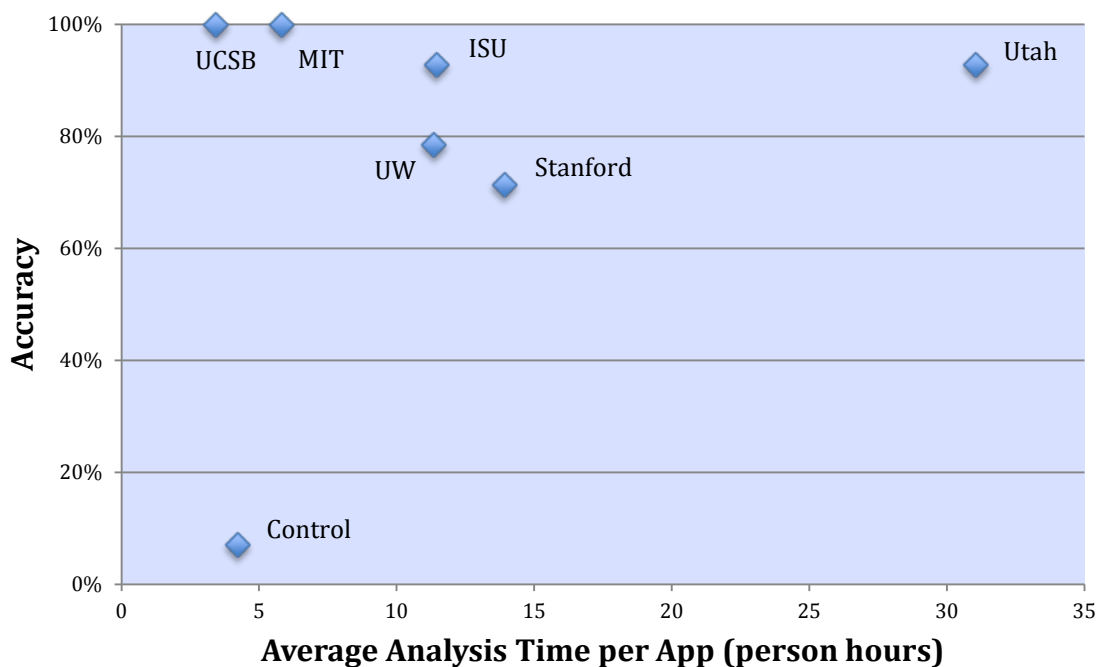


Figure 10 - Accuracy vs. Analysis Time for Experiment 4A

Figure 11 displays a comparison of the average accuracy scores for each analyst team during Experiments 3A and 4A. All of the R&D teams improved their accuracy scores during Experiment 4A. The greatest increases were from ISU who increased their accuracy from 60% to 93%, and from UW who improved their accuracy from 30% to 79%. The accuracy score for the Control Team decreased from the previous experiment.

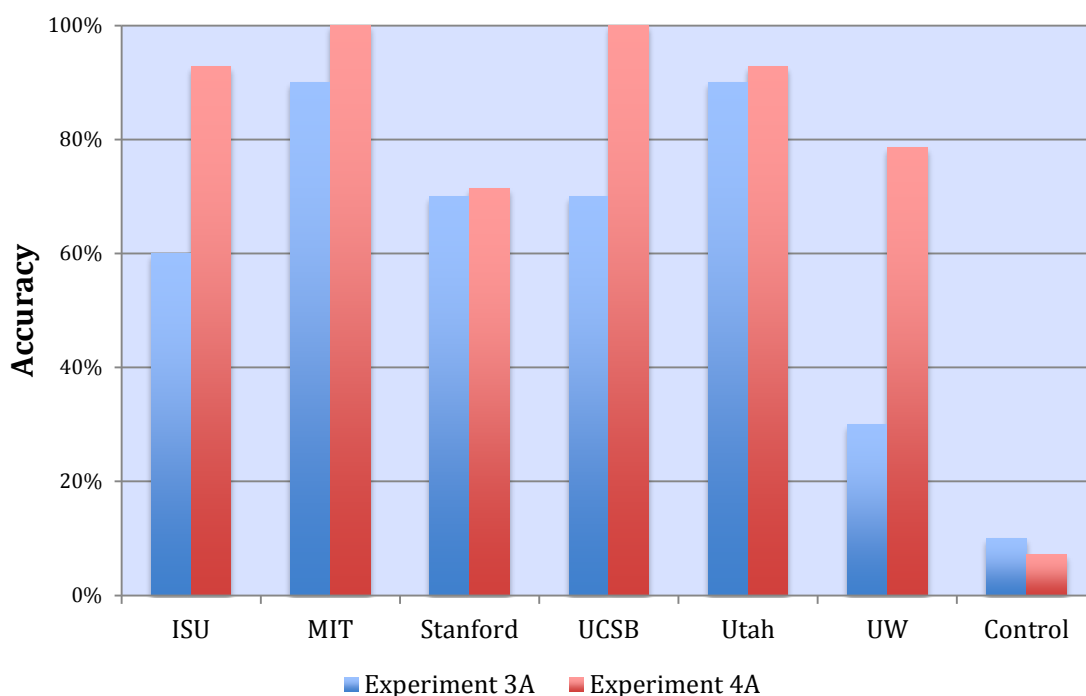


Figure 11 - Accuracy Comparison for Experiments 3A and 4A

Figure 12 displays a comparison of the average analysis times for each analyst team during Experiments 3A and 4A. All of the analyst teams, except for MIT, increased their analysis times when compared to the previous experiment. Utah increased their average analysis time dramatically due to spending over 100 hours analyzing two applications. Notably, MIT was able to reduce their analysis time by more than 50% during Experiment 4A while maintaining a 100% accuracy score.

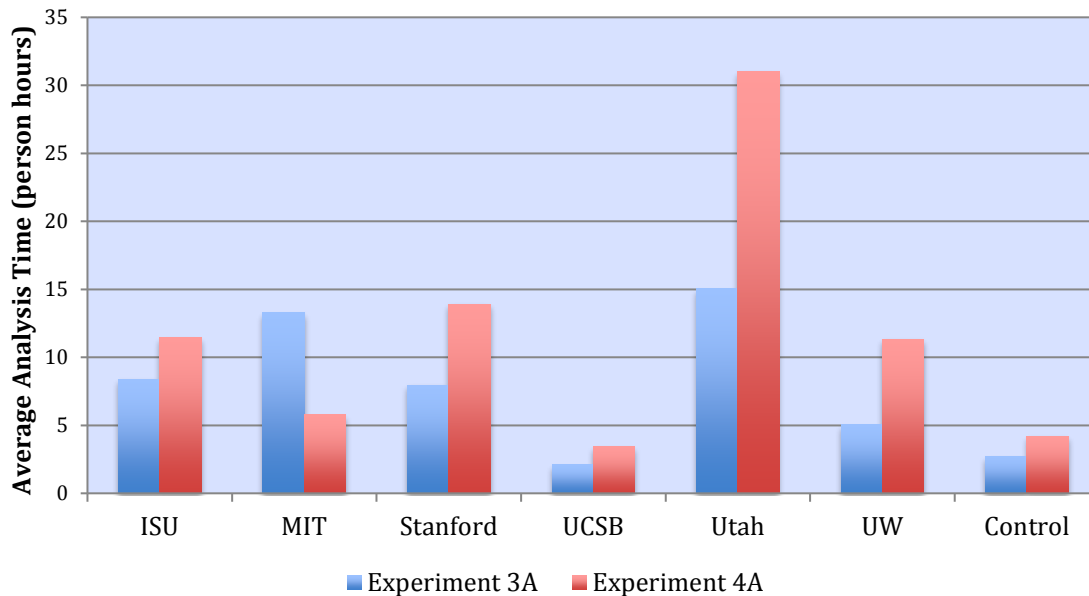


Figure 12 - Analysis Times for Experiments 3A and 4A

Figure 13 displays the average analysis time for each application in Experiment 4A. Again, the exaggerated analysis times from Utah cause the times for Memotis and Vermilion-Update3 to be significantly longer than other applications. The applications with the shortest average analysis times were CodeGen and Creations. These two applications each required approximately three hours to analyze and every R&D team analyzed them correctly. All of the remaining applications required five to fifteen hours of analysis time on average. These average analysis times can be used as a rough estimate for how long the APAC tools currently require to correctly analyze an application.

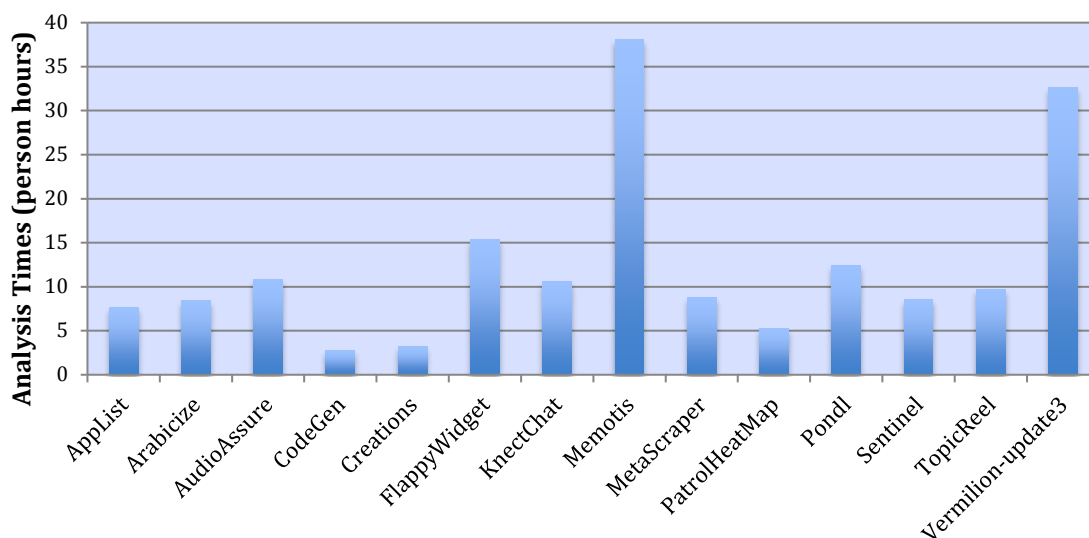


Figure 13 - Experiment 4A Average Analysis Times per App

### 4.8.2 The Take-Home Oracle

The Oracle system was successful in keeping the R&D teams searching for the intended malware during Experiment 4A. There were almost two hundred distinct Oracle email queries during the experiment. Figure 14 shows the number of Oracle queries from each team for each day of the experiment.

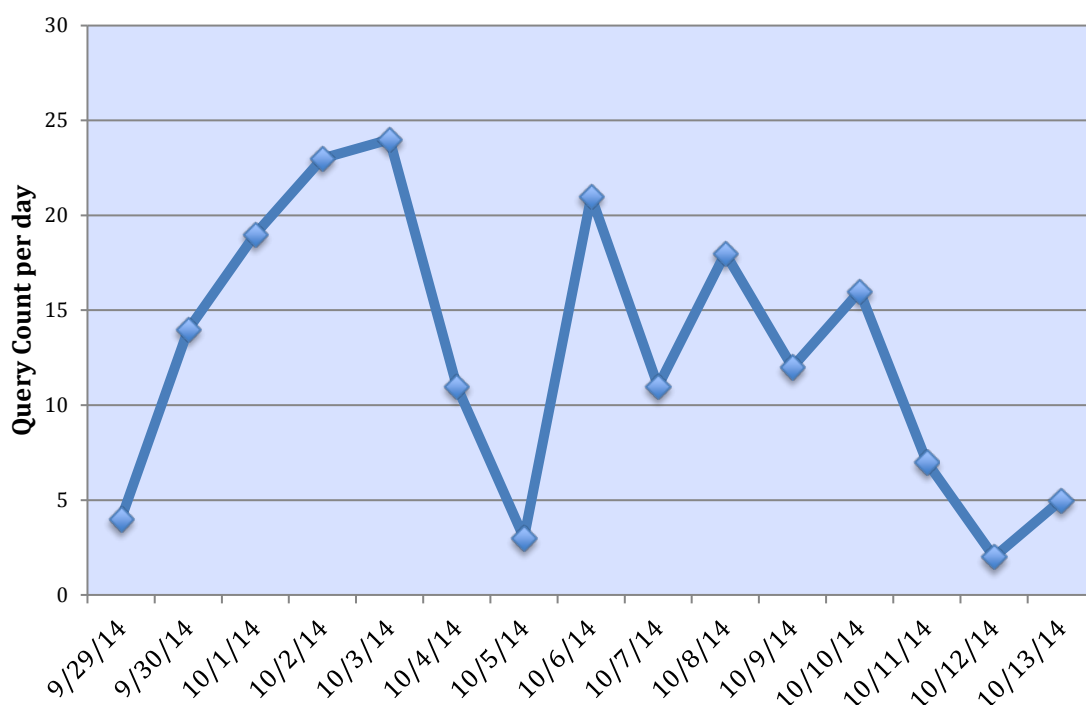


Figure 14 - Experiment 4A Oracle Queries by Date

### 4.8.3 Control Team Performance

The Control Team's performance during Experiment 4A was disappointing. Of the fourteen malicious applications, the Control Team only provided result documents for seven. The Control Team consulted the Oracle for only one of the malicious applications. This was also the only application that they analyzed correctly. An email was sent to the Control Team asking if they had analyzed the other seven applications and why they had not made greater use of the Oracle. No response was received.

It is important for the Control Team to improve their performance for future APAC experiments. Without reliable results from the Control Team it will be difficult to compare their results to those of the R&D teams.

## 4.9 Engagement 5

Engagement 5 consisted of two experiments, a take-home experiment and an onsite experiment during the APAC PI meeting in late April. The first experiment, Experiment 5A, was a two-week take-home experiment that began on March 23<sup>rd</sup>, 2015 and ended on April 6<sup>th</sup>, 2015. This experiment contained twelve malicious applications to be investigated by the analyst teams. These teams consisted of five APAC R&D teams using their custom developed analysis tools. Unlike previous engagements, a control team was not utilized during Engagement 5. As in previous experiments, the Oracle question and answer system was in effect for Experiment 5A.

The length of this take-home experiment, and the number of applications used, was influenced by the results of Engagement 4. The take-home experiment for Engagement 4 also had a duration of two-weeks (including weekends). This duration allowed the R&D teams greater scheduling flexibility when analyzing applications. All of the R&D teams were able to analyze the applications during this period, with UCSB actually completing the experiment during the first week.

### 4.9.1 Experiment 5A Results

The results of Experiment 5A showed a continued trend of increasing accuracy for the tools being developed for the APAC program. Here are the key points:

- MIT, UCSB, and Utah found the malware in all applications
- UCSB had the shortest average analysis times, and completed the experiment at the end of the first week
- MIT had the next shortest analysis time, while Utah had the longest analysis time of all teams
- Stanford found 75% of the malware, and UW found 67%

The number of R&D teams that were able to correctly analyze all of the malicious applications increased to three during Experiment 5A, whereas only two teams were able to achieve this in Experiment 4A. These three teams (MIT, UCSB, and Utah) can be differentiated by their widely varying analysis times.

Figure 15 displays the average accuracy and analysis time results for each performer in Experiment 5A. This graph plots the results based on how many applications each team can analyze during an eight hour time period.

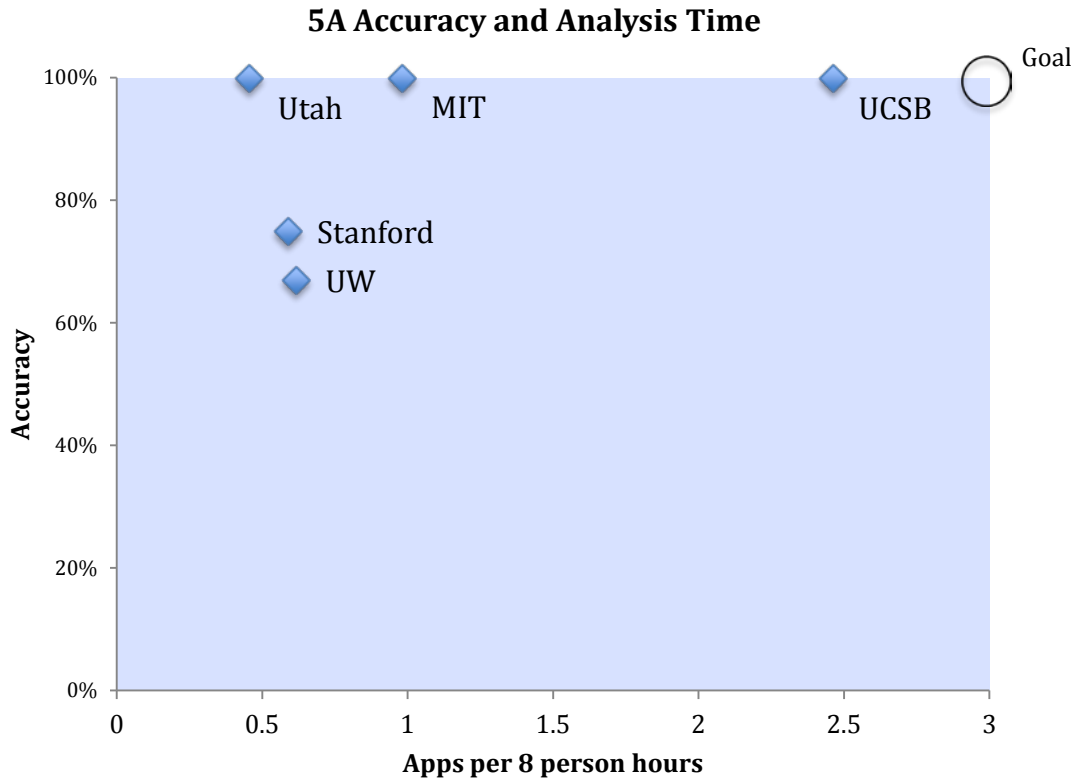


Figure 15 – Accuracy and Analysis Time for Experiment 5A

As indicated, the goal in this graph is to have the highest accuracy score while also having the highest number of applications analyzed during the eight hour time period. UCSB is a clear standout in these results, with an analysis time that is over two times faster than their closest competitor.

Figure 16 shows a breakdown of the analysis times for each performer during Experiment 5A. This graph shows the contribution of the human analysis times, configuration times, and automated analysis times to the overall analysis time for each team. In all cases, the human analysis time is the largest component of the overall analysis times. Interestingly, automated analysis time is an insignificant factor for most of the teams, with MIT and UCSB (to a lesser extent) being the only exceptions.

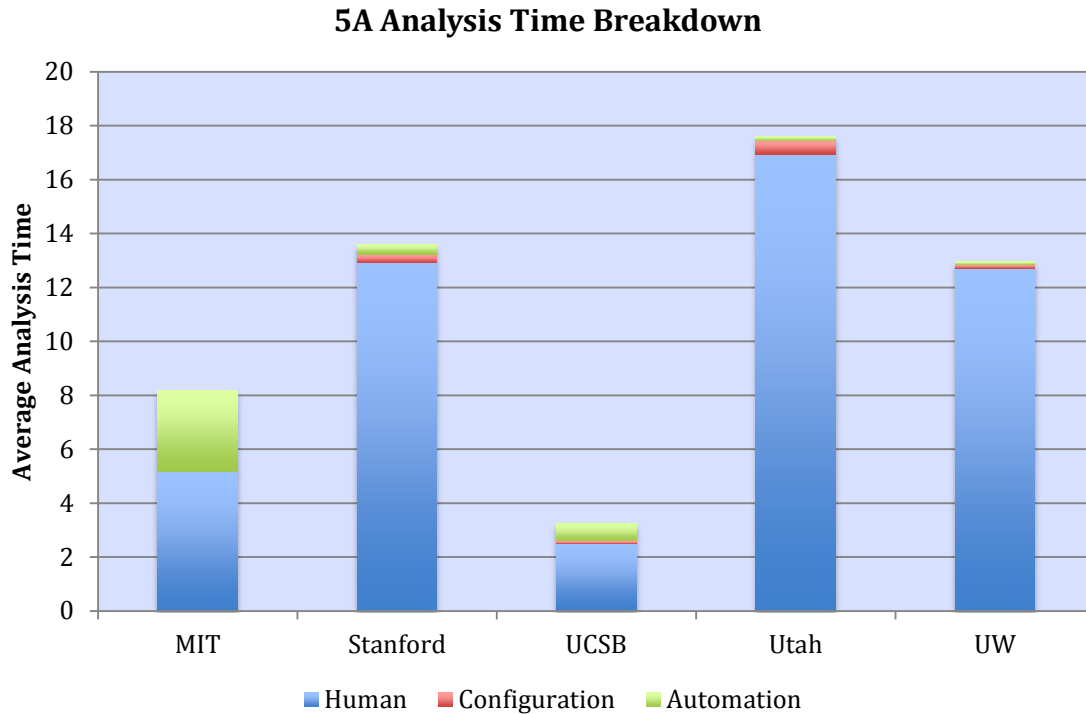


Figure 16 - Analysis Time Breakdown for Experiment 5A

Figure 17 displays a comparison of the average accuracy scores for each analyst team during Experiments 3A, 4A and 5A. All but one of the R&D teams (UW) continued to either improve or maintain their accuracy scores during Experiment 5A.

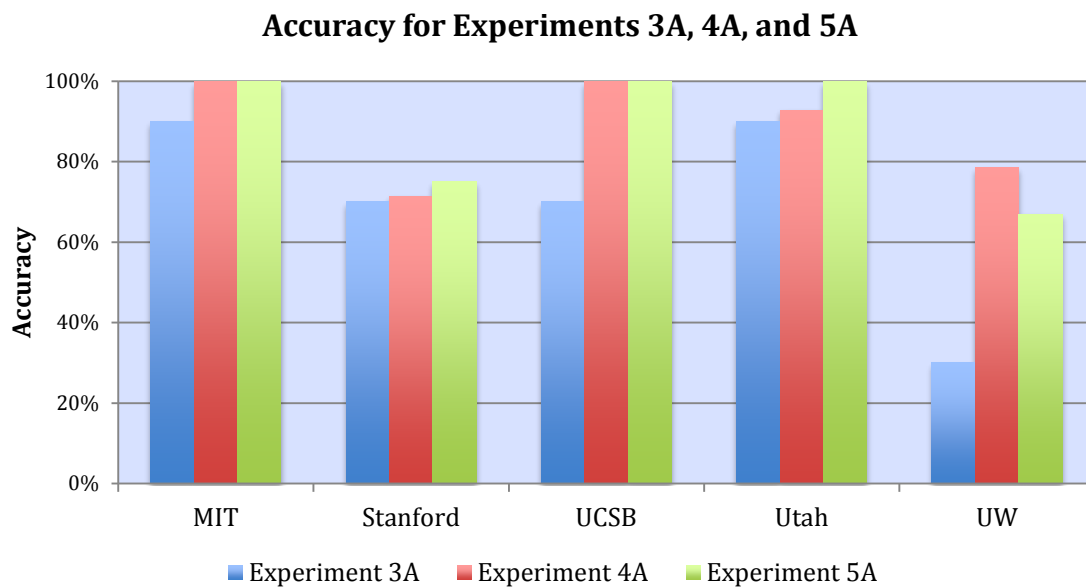


Figure 17 - Accuracy Comparison for Experiments 3A, 4A, and 5A

Figure 18 displays a comparison of the average analysis times for each analyst team during Experiments 3A, 4A, and 5A. In general, most of the teams' analysis times were similar to their previous performance. The one standout was Utah, who was not only capable of correctly analyzing all applications, but was also able to dramatically improve their analysis time compared to their result from Experiment 4A.

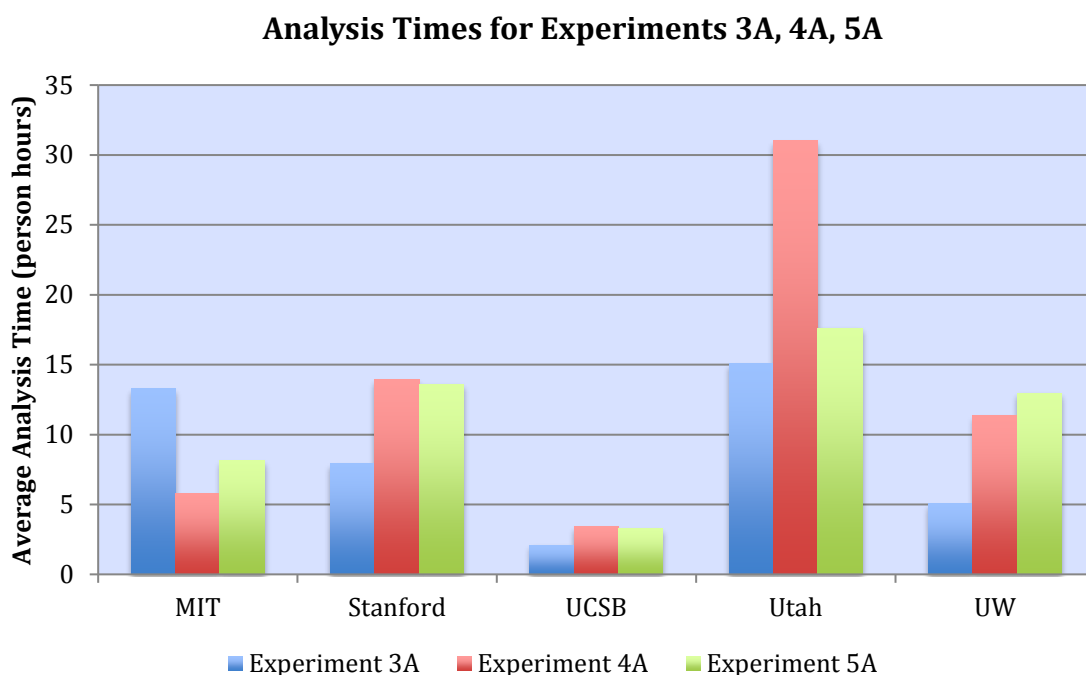


Figure 18 - Analysis Times for Experiments 3A, 4A, and 5A

Figure 19 displays a combined graph of the average accuracy and analysis times for each application in Experiment 5A. The blue bar chart indicates the accuracy for each application, and the red line graph indicates the analysis times. Based on these results, it can be seen that many of the applications were correctly analyzed by all of the teams. The two least accurately analyzed applications were ConferenceMaster and Noiz2, with ConferenceMaster also having the highest average analysis time of all applications.

The stated purpose of ConferenceMaster was to be a conference call auto-dialer that recorded the audio tones of conference call phone numbers and passcodes. The malicious functionality was that under certain circumstances, an attacker controlled phone number would be prepended to outgoing calls, allowing for call interception and monitoring. In these cases, the displayed phone number would appear correct to the user. However, the emitted audio tones would include the attacker controlled number. Some of the teams were able to analyze ConferenceMaster in a short time period, while others spent dramatically more time. The malware in this application was straightforward to reproduce dynamically, so that may have helped some teams to better focus their analysis efforts. ConferenceMaster contained over 15000 lines of code. However, only

10% of that code was application specific, with the remainder being included library source code.

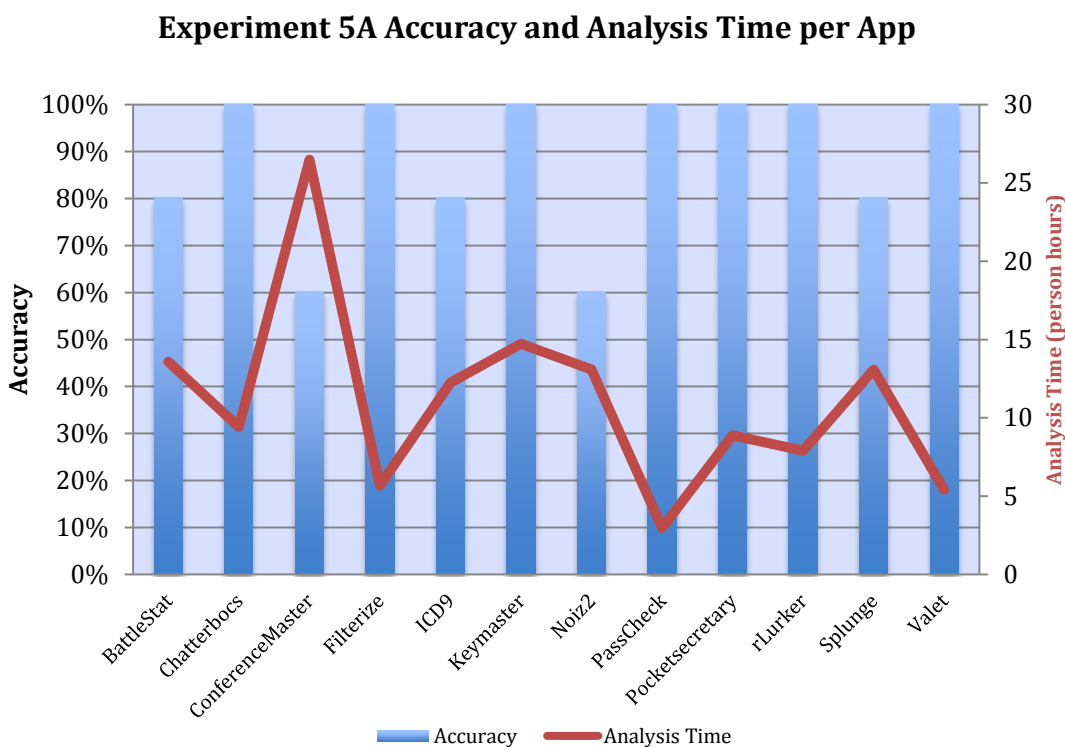


Figure 19 - Experiment 5A App Accuracy and Analysis Time

Noiz2 was a video game that included local network player vs. player functionality. The malicious behavior was that Noiz2 secretly performed network reconnaissance (disguised as network discovery of other Noiz2 players) and allowed a remote attacker to retrieve the collected network data. The intended attack scenario was for a Noiz2 user to connect to a sensitive network (such as their corporate network) after having played the game on a more innocuous network such as the local coffee shop. This would allow an attacker to collect network data from the sensitive network without requiring direct access to it.

#### 4.9.2 The Take-Home Oracle

The Oracle system was successful in keeping the R&D teams searching for the intended malware during Experiment 5A. There were fewer than one hundred distinct Oracle email queries during the experiment. This number was down significantly from the previous take-home experiment that had almost two hundred queries. This decrease can be partially explained by Experiment 5A only having five participating teams whereas Experiment 4A had seven teams. There were also two fewer applications used during Experiment 5A. Even when accounting for these differences, there was still a reduction in the use of the Oracle.

The majority of the queries during Experiment 5A were for correct detections. Only around thirty of the emails were for unintended malware detections, responses to requests for further details, or general support for compiling the applications. Figure 20 shows a graph of the number of Oracle queries for each day of the experiment.

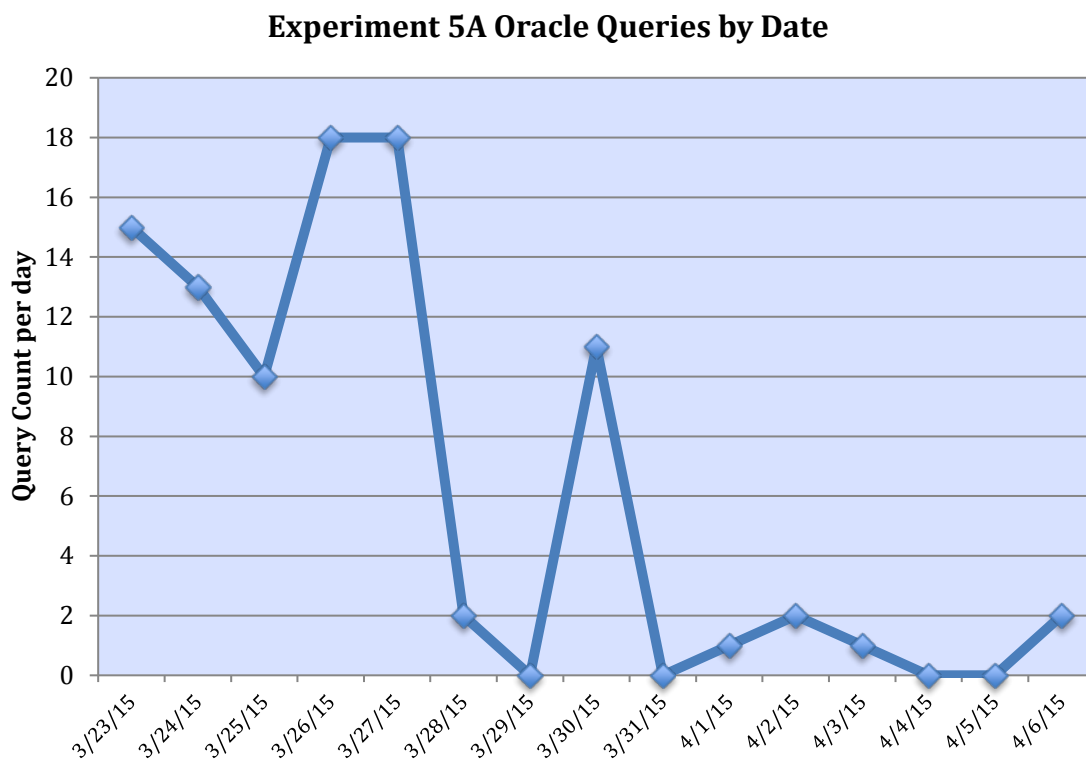


Figure 20 - Experiment 4A Oracle Queries by Date

#### 4.9.3 Experiment 5B

Prior to the beginning of Experiment 5A, it was decided that an onsite experiment was needed during the April PI meeting. To meet this need, Experiment 5B was created. This experiment was modeled off of the previous onsite experiment (3B) that took place during the April 2014 PI meeting at Carnegie Mellon University. During that experiment, the analyst teams were given four malicious applications to analyze during a five hour time period. Experiment 3B proved to be a difficult challenge for the analysts, with the majority of the teams only able to make one correct detection during the experiment.

In order to provide more useful results, Experiment 5B was adjusted to include only three malicious applications. The teams were allowed to use five analysts, and the duration of the experiment was increased to six hours and thirty minutes (including a working lunch). The applications that were selected for the experiment included two applications that were removed from Experiment 5A, and one unused application from a previous engagement.

Teams that required potentially long running automated analysis were given early access to the challenge applications for pre-computation. To support this, Five Directions provided a virtual machine server where participating teams could upload their tools and start their automated analysis. Teams using pre-computation were instructed not to manually analyze the applications during this time. Only MIT and UCSB took advantage of pre-computation.

As with the previous onsite experiments, the Oracle system was in effect for Experiment 5B.

#### 4.9.4 Experiment 5B Results

The adjustments that were made to Experiment 5B proved to be effective. The majority of the teams required a significant amount of the experiment duration in order to analyze the applications, with two teams (Utah and UCSB) using the entire available time. The standout performance during Experiment 5B was Stanford. Stanford was able to correctly analyze each application, and they only required two hours and forty minutes to complete the entire experiment.

Three other teams (MIT, UCSB, and UW) were also able to correctly analyze each application. As with Experiment 5A, the performance of these teams can be differentiated based on their analysis times. Utah correctly analyzed two of the three malicious applications. They were close to correctly analyzing the final application, but they ran out of time at the end of the experiment.

Figure 21 displays the average accuracy and analysis time results for each performer in Experiment 5B. This graph plots the results based on how many applications each team could analyze during a four hour time period.

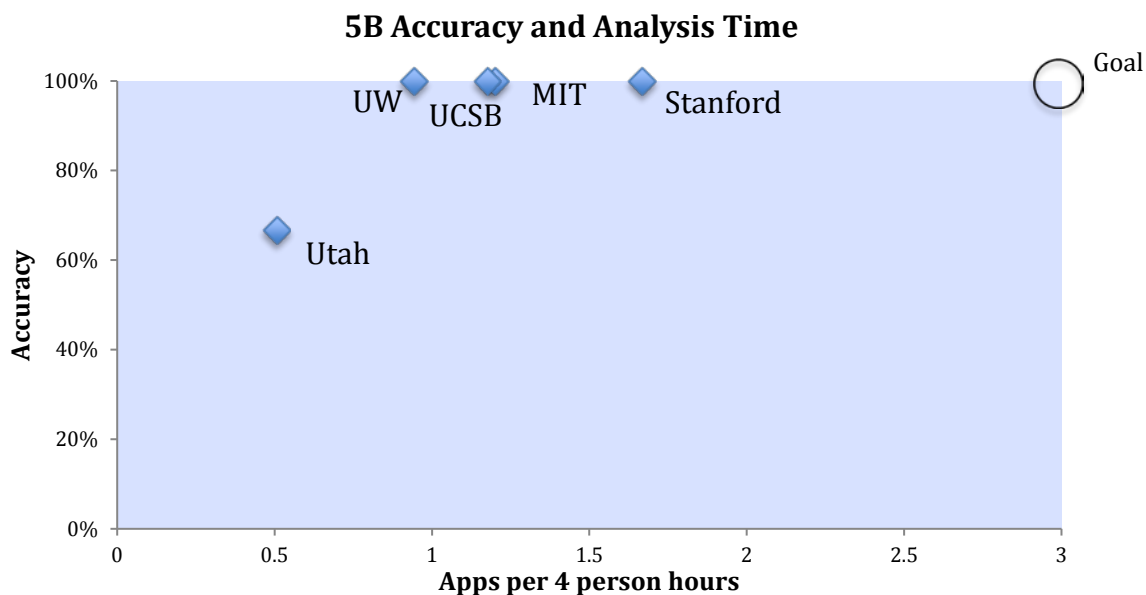


Figure 21 – Accuracy and Analysis Time for Experiment 5B

The onsite experiments allow for a direct observance of the total number of analysts and the total analysis time used by each team. Table 4 lists the number of analysts each team used, their total self-reported time, and their total observed time. The self-reported time is the sum of the analysis times each team reported for each application. The observed time is calculated by multiplying the number of analysts for each team by the total time they used during the experiment (as observed by Five Directions). The time that each team handed in their results was used as the total time for each team. As an example of calculating the observed time, Stanford had four analysts and completed the experiment after two hours and forty minutes. This results in a total observed time of ten hours and 36 minutes (4 analysts x 160 minutes = 10.6 hours).

**Table 4 - Experiment 5B Self-Reported vs. Observed Time (hours)**

<b>Team</b>	<b># of analysts</b>	<b>Self-Reported Total</b>	<b>Observed Total</b>
<b>MIT</b>	5	10*	27.9
<b>Stanford</b>	4	7.2	10.6
<b>UCSB</b>	5	10.2*	32.5
<b>Utah</b>	5	23.7	32.5
<b>UW</b>	4	12.7	25

\*includes pre-computation time

The observed total times can be used as a ceiling on the analysis times that each team required to analyze the malicious applications. The discrepancy between the observed times and the self-reported times can be partially explained by teams using the experiment time to write their result documents. It is also possible that each analyst was not analyzing during the entire time their team was in the experiment room. For example, from the Oracle queries reported to Five Directions, UCSB correctly analyzed all of the applications by early afternoon, but they did not hand in their results until the end of the experiment time. Therefore their observed total was longer.

Figure 22 shows a breakdown of the analysis times for each performer during Experiment 5B. This graph shows the contribution of the human analysis times, configuration times, and automated analysis times to the overall analysis time for each team. As in Experiment 5A, the human analysis time is the largest component of the overall analysis times.

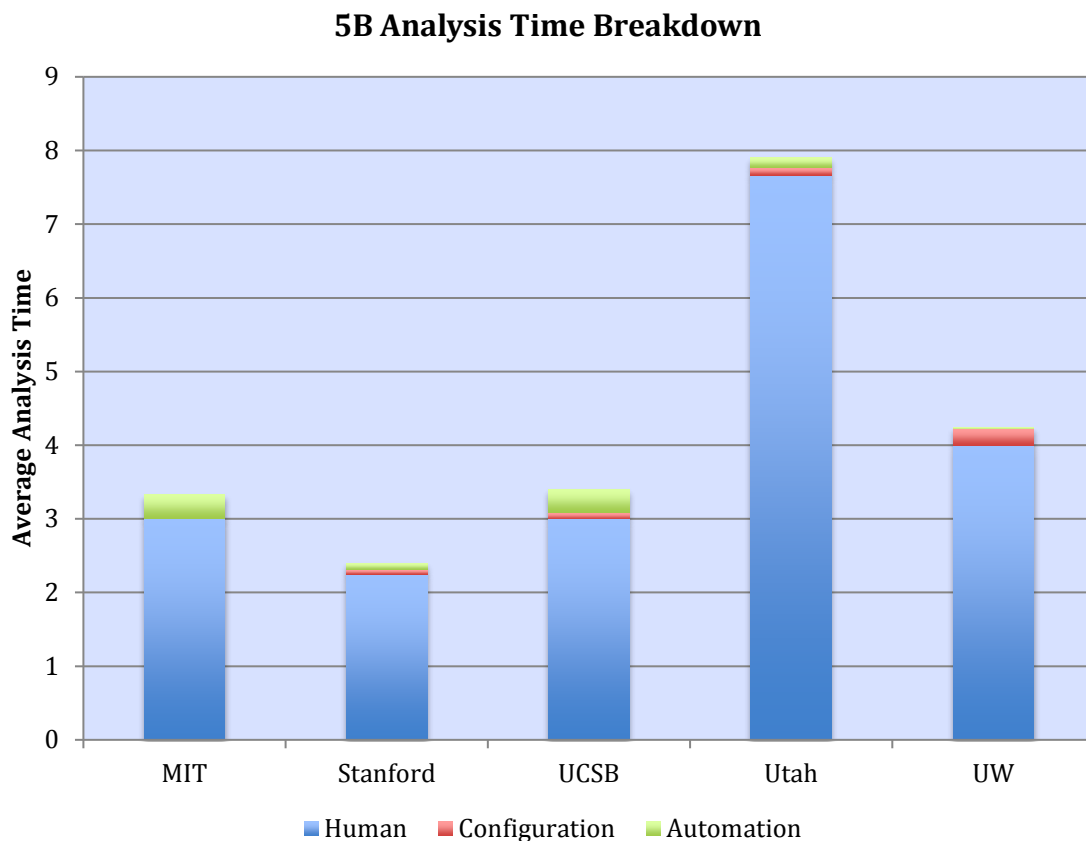


Figure 22 - Analysis Time Breakdown for Experiment 5B

Figure 23 shows the average analysis time for each application from Experiment 5B. The RFDrop application proved to be the most difficult application to analyze. It had the longest analysis time and was the only application not to be correctly analyzed by all teams. The stated purpose of RFDrop was to allow covert agents to exchange encrypted messages simply by walking past each other. The application would use the radio transceiver in each phone to send the messages without any direct physical contact. RFDrop could also be configured as a dead-drop device where agents could transfer messages at different times by walking past the device. The maliciousness in RFDrop was that when configured as a dead-drop, the application would transmit the encryption key in plain text in a separate message along side the intended message. This would allow an attacker to decrypt the messages being sent by the agents. This may have been difficult to analyze since the attack vector used the same communication channel as the sensitive encrypted messages.

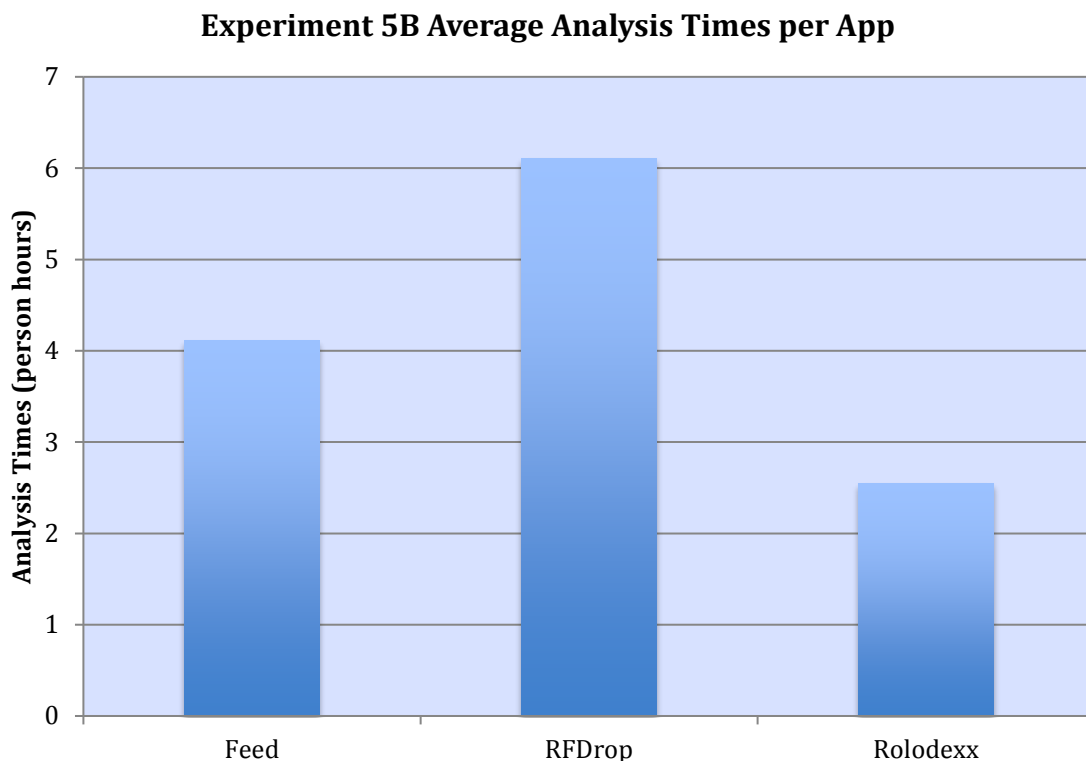


Figure 23 - Average Analysis Time per App for Experiment 5B

#### 4.10 Engagement 6

Engagement 6 consisted of two experiments, a take-home experiment and an onsite experiment during the APAC PI meeting in late October. The first experiment, Experiment 6A, was a one-week take-home experiment that began on September 28<sup>th</sup>, 2015 and ended on October 5<sup>th</sup>, 2015. This experiment contained eight malicious applications to be investigated by the analyst teams. These teams consisted of five APAC R&D teams using their custom developed analysis tools. As in the previous engagement, a control team was not utilized during Engagement 6. The Oracle question and answer system was in effect for Experiment 6A.

The length of this take-home experiment was reduced to one week (including the weekend) based on the number of applications delivered by the AC team. The AC team delivered a total of twelve malicious applications. These twelve applications were split into eight applications for the take-home experiment, and four applications for the on-site experiment. The take-home experiment for Engagement 5 had a duration of two-weeks (including weekends) where the performers analyzed twelve applications. Since the number of applications for the Experiment 6A was reduced to eight, the duration was also reduced.

#### 4.10.1 Experiment 6A Results

The results of Experiment 6A showed that the trend of increasing accuracy for the tools being developed for the APAC program continues. Here are the key points:

- MIT, UCSB, and Utah found the malware in all applications
- UCSB had the shortest average analysis time, Utah had the longest
- Stanford found 75% of the malware
- UW correctly analyzed one application

The number of R&D teams that were able to correctly analyze all of the malicious applications continued to be three during Experiment 6A. These three teams (MIT, UCSB, and Utah) can be differentiated by their widely varying analysis times.

Figure 24 displays the average accuracy and analysis time results for each performer in Experiment 6A. This graph plots the results based on how many applications each team can analyze during an eight hour time period.

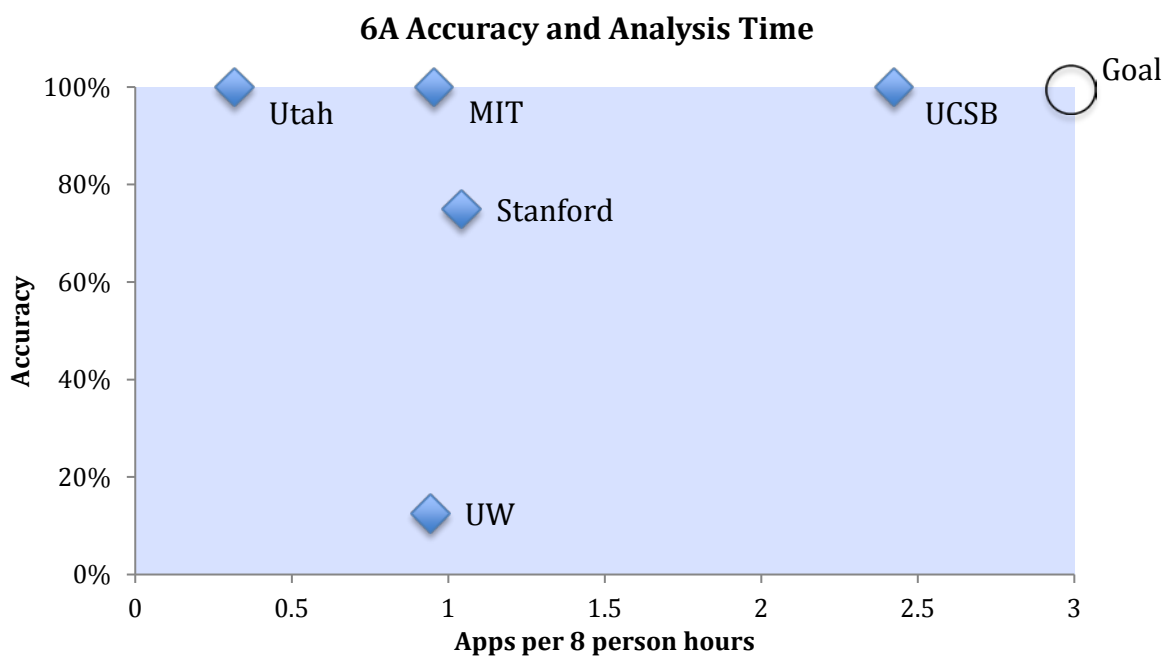


Figure 24 – Accuracy and Analysis Time for Experiment 6A

As indicated, the goal in this graph is to have the highest accuracy score while also having the highest number of applications analyzed during the eight hour time period. UCSB is a clear standout in these results, with an analysis time that is over two times faster than their closest competitor.

Figure 25 shows a breakdown of the analysis times for each performer during Experiment 6A. This graph shows the contribution of the human analysis times, configuration times, and automated analysis times to the overall analysis time for each team. In almost all cases, the human analysis time is the largest component of the overall analysis times. MIT is the exception, where their automated analysis time is longer than their human analysis time.

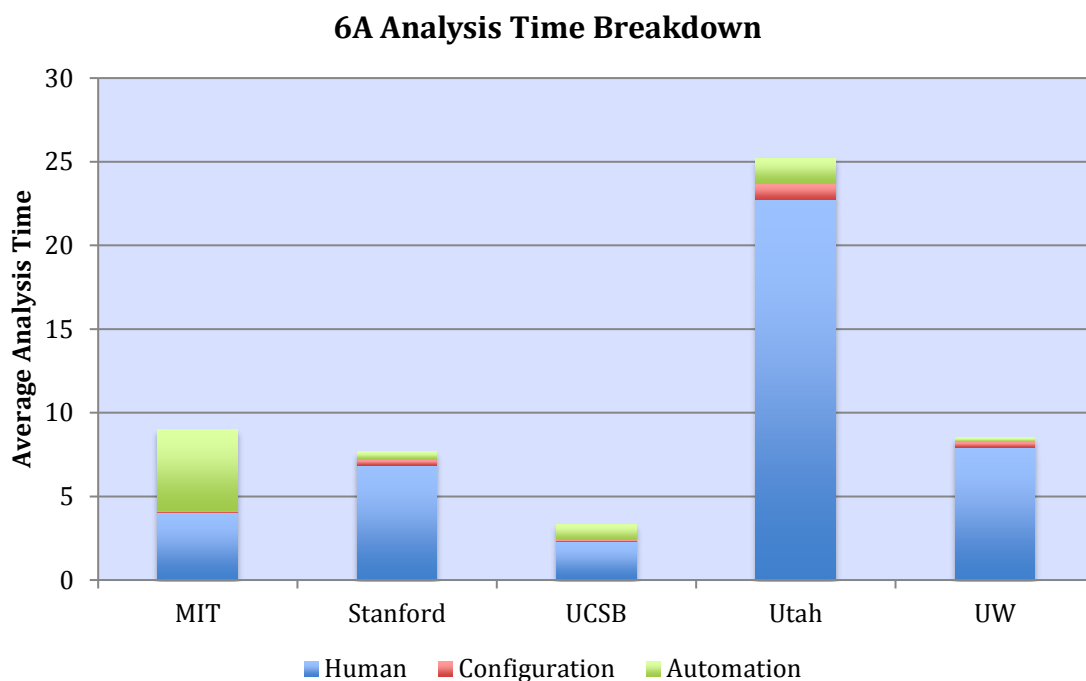


Figure 25 - Analysis Time Breakdown for Experiment 6A

Figure 26 displays a comparison of the average accuracy scores for each analyst team during Experiments 3A, 4A, 5A, and 6A. All but one of the R&D teams (UW) continued to either improve or maintain their accuracy scores during Experiment 6A.

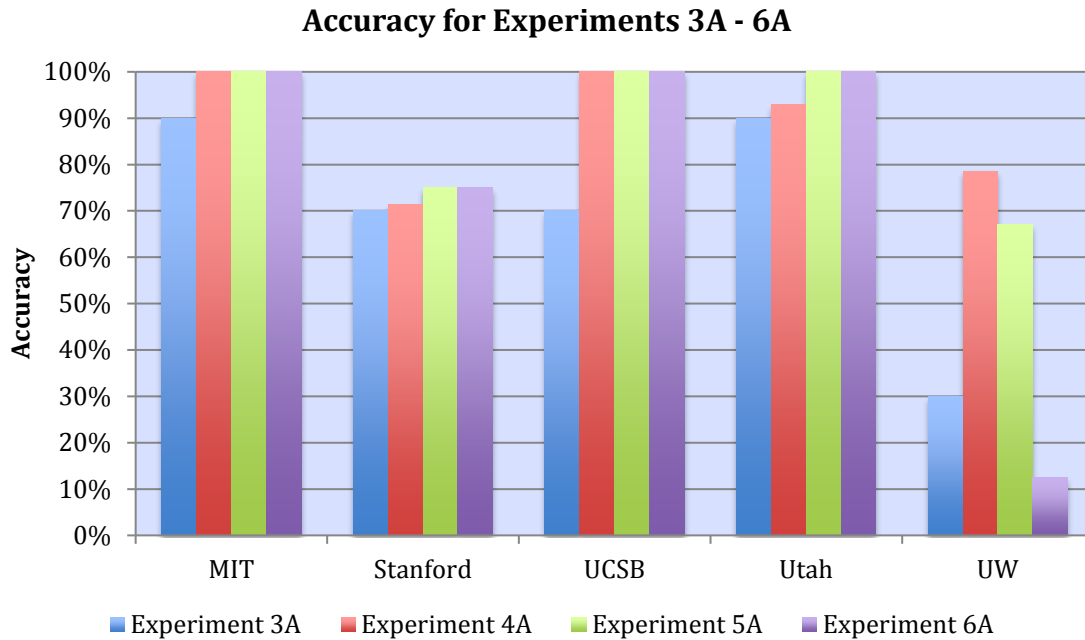


Figure 26 - Accuracy Comparison for Experiments 3A - 6A

Figure 27 displays a comparison of the average analysis times for each analyst team during Experiments 3A, 4A, 5A, and 6A. In general, most of the teams' analysis times were similar to their previous performance.

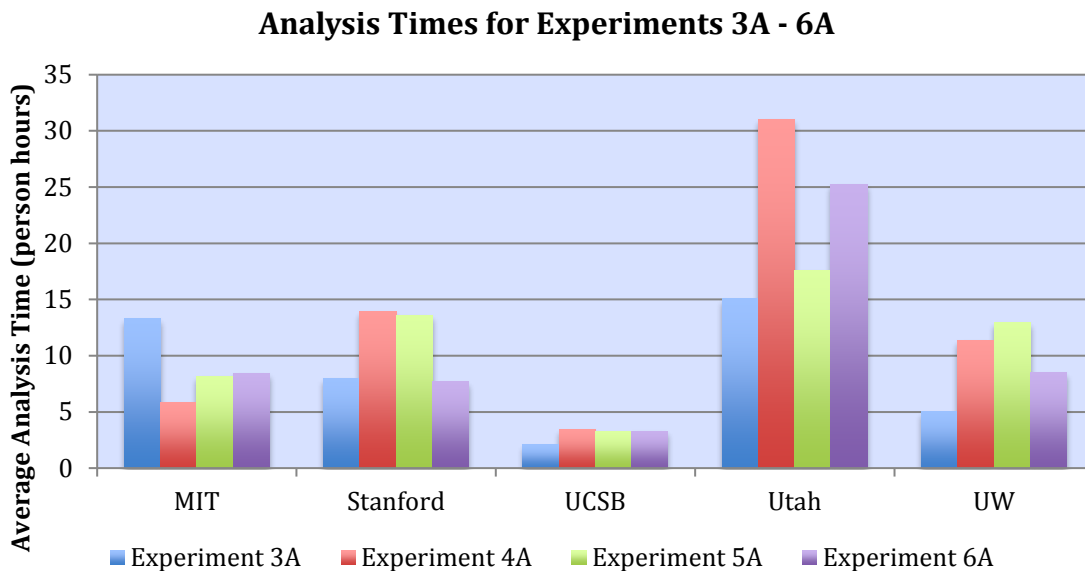


Figure 27 - Analysis Times for Experiments 3A - 6A

Figure 28 is a graph of analysis times for each malicious application in Experiment 6A. This graph shows the average and median analysis times for each application used in the experiment.

The Koenigsberg application had the longest average analysis time. However, a single large analysis time distorted that average, as can be seen by its median analysis time.

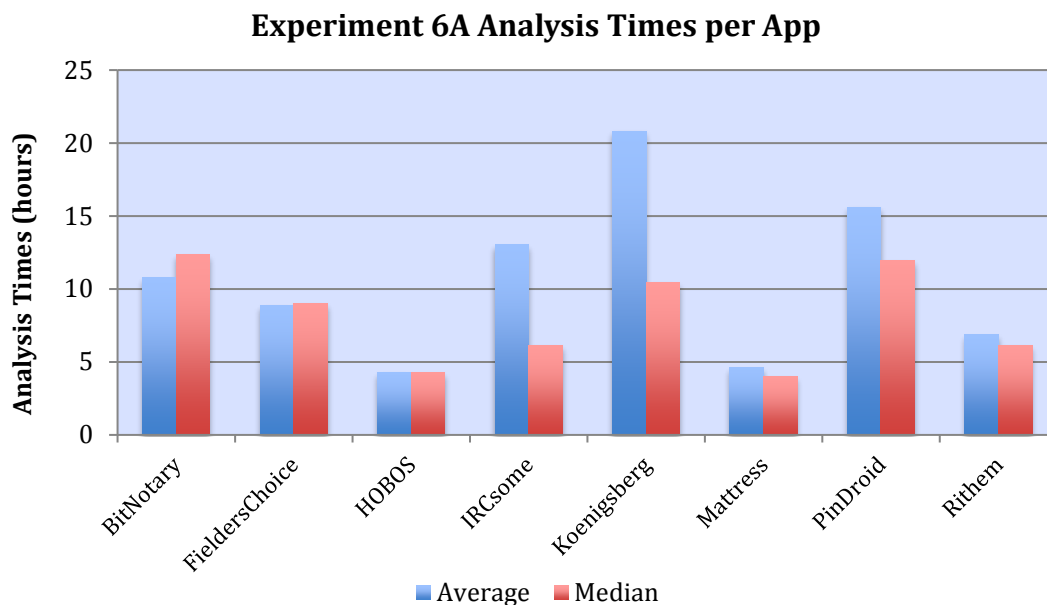


Figure 28 - Experiment 6A Analysis Time per App

#### 4.10.2 The Take-Home Oracle

The Oracle system was successful in keeping the R&D teams searching for the intended malware during Experiment 6A. There were fewer than seventy distinct Oracle email queries during the experiment.

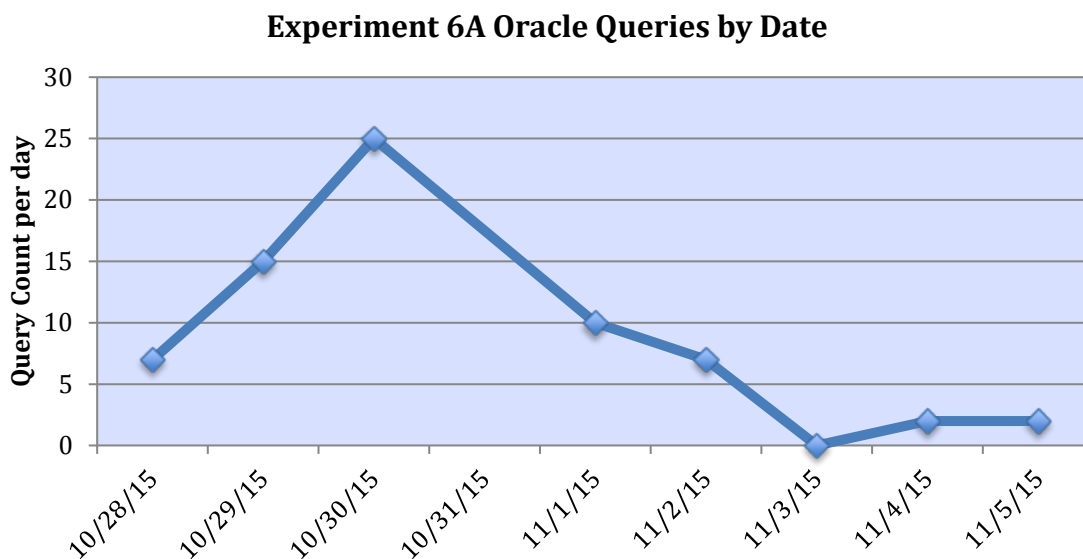


Figure 29 - Experiment 6A Oracle Queries by Date

#### 4.10.3 Experiment 6B

Experiment 6B was an on-site experiment that was held during the October APAC PI meeting. During this experiment the performers analyzed four malicious applications during a six hour and thirty minute session.

Teams that required potentially long-running automated analysis were given early access to the challenge applications for pre-computation. To support this, Five Directions provided a virtual machine server where participating teams could upload their tools and start their automated analysis. Teams using pre-computation were instructed not to manually analyze the applications during this time. MIT, UCSB, and UW took advantage of pre-computation.

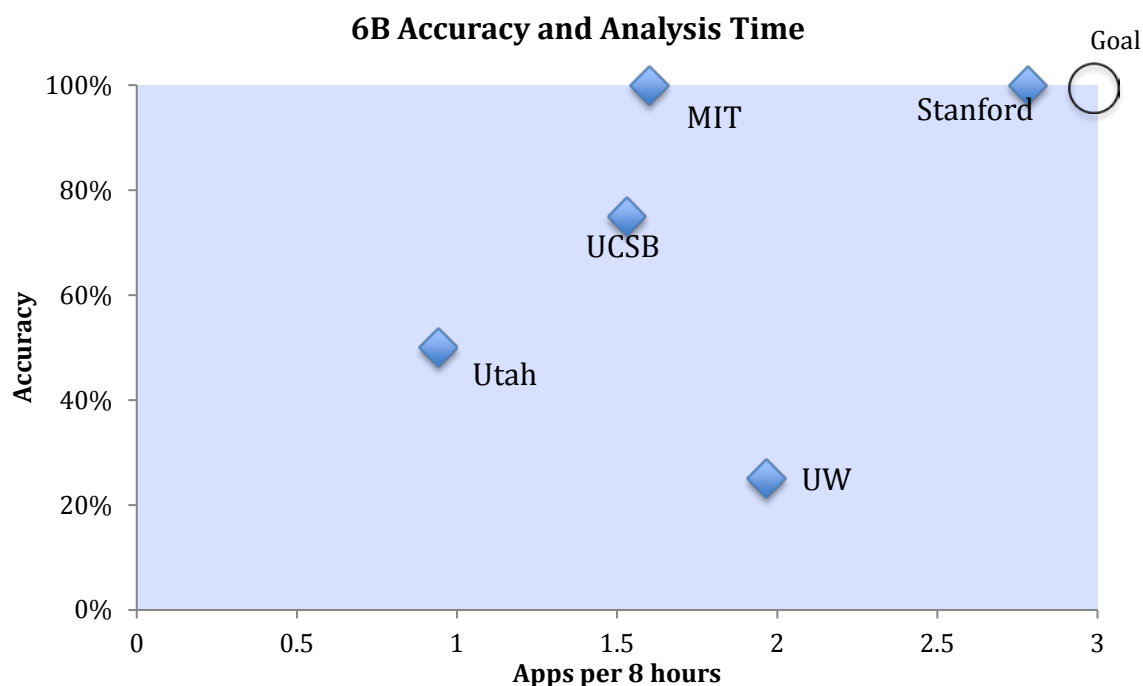
As with the previous onsite experiments, the Oracle system was in effect for Experiment 6B.

#### 4.10.4 Experiment 6B Results

Here are the key points from Experiment 6B:

- MIT and Stanford found the malware in all applications
- Stanford had the shortest average analysis time, Utah had the longest
- UCSB found 75% of the malware
- Utah found 50% of the malware
- UW correctly analyzed one application

Figure 30 displays the average accuracy and analysis time results for each performer in Experiment 6B. The results are plotted based on how many applications each team could analyze during a four hour time period.



**Figure 30 – Accuracy and Analysis Time for Experiment 6B**

The onsite experiments allow for a direct observance of the total number of analysts and the total analysis time used by each team. Table 5 lists the number of analysts each team used, their total self-reported time, and their total observed time. The self-reported time is the sum of the analysis times each team reported for each application. The observed time is calculated by multiplying the number of analysts for each team by the total time they used during the experiment (as observed by Five Directions). The time that each team submitted their results was used as the total time.

**Table 5 - Experiment 6B Self-Reported vs. Observed Time**

<b>Team</b>	<b># of analysts</b>	<b>Self-Reported Total</b>	<b>Observed Total</b>
<b>MIT</b>	4	19.9*	26
<b>Stanford</b>	4	11.5	26
<b>UCSB</b>	4	20.9*	26
<b>Utah</b>	4	34	26
<b>UW</b>	3	16.2*	19.5

\*includes pre-computation time

The observed total times can be used as a ceiling on the analysis times that each team required to analyze the malicious applications. The discrepancy between the observed times and the self-reported times can be partially explained by teams using the experiment time to write their result documents. It is also possible that each analyst was not analyzing during the entire time their team was in the experiment room. Utah's self-reported time is actually longer than their observed time. All of their result documents listed 8.5 hours of analysis time for each application. It is possible that their self-reported times are either inaccurate, or that they had automated analysis running in parallel with the human analysis.

Figure 31 shows a breakdown of the analysis times for each performer during Experiment 6B. This graph shows the contribution of the human analysis times, configuration times, and automated analysis times to the overall analysis time for each team. In this experiment the automated analysis times were not a significant portion of the overall analysis times.

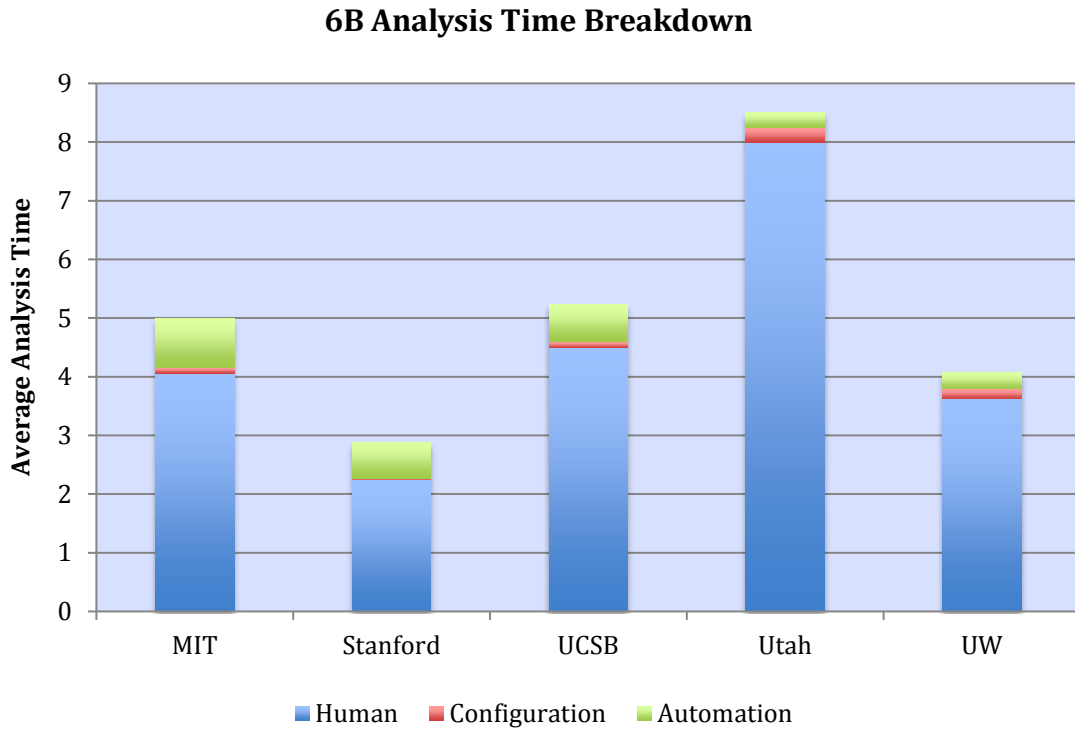


Figure 31 - Analysis Time Breakdown for Experiment 6B

Figure 32 shows the average analysis time for each application from Experiment 6B.

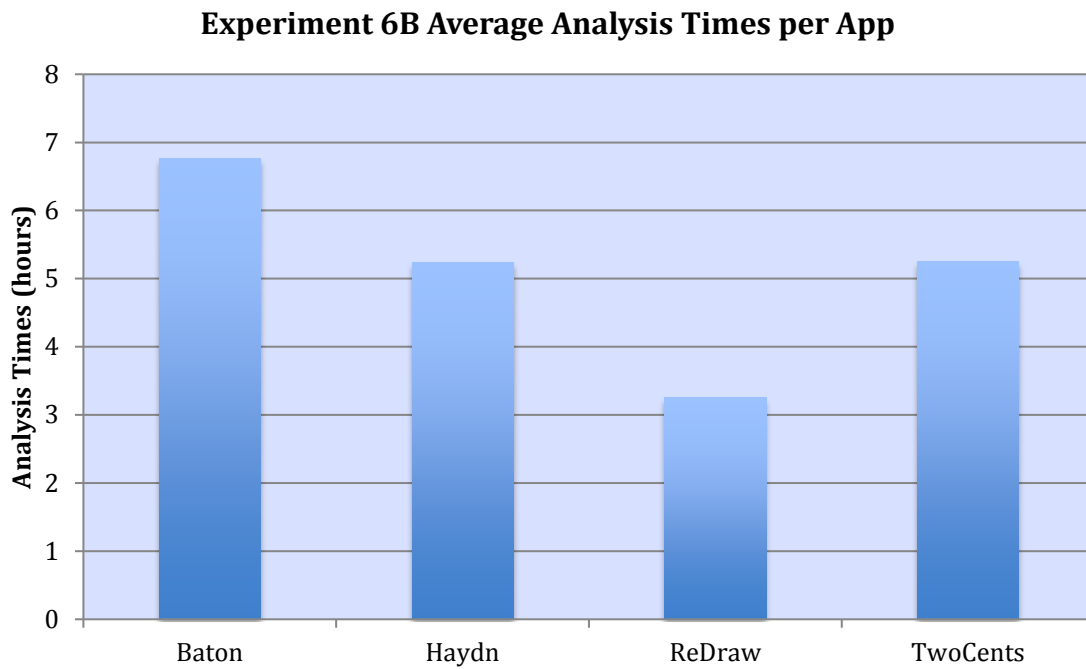


Figure 32 - Average Analysis Time per App for Experiment 6B

Experiment 6B was the final adversarial experiment for the APAC program.

## 5 CONCLUSIONS

### 5.1 Oracle Question and Answer System

During the early experiments in Engagement 1 and 2, if the teams reported issues in the malicious applications that could be reasonably considered malware, then they were awarded correct detections. Since they had no way of knowing if this was the intended malware or not, this led to the situation where they would stop analysis when finding relatively simple issues, and miss intended malware that was more difficult to find.

In an effort to prevent teams from short-circuiting their analysis on an application when finding suspicious behavior that was not the intended malware, we implemented an ‘oracle’ question and answer system starting with Experiment 2B. This system allowed the teams to email Five Directions during the experiment when they found something they considered malware. The ‘oracle’ would then inform them if they had found the correct malware or if they should continue analysis.

There were numerous oracle emails received during the Experiments and this system prevented the teams from terminating their analysis prematurely. It forced them to continue looking for the more elusive malware and therefore gave a more accurate picture of the detection capabilities of the tools.

Feedback from Engagement 3 led to a modification of the possible responses given by the Oracle. In previous engagements, the Oracle would provide only two responses: “You can stop analyzing this app now” or “Keep looking”. One potential drawback to these responses was that a team might be on the right track in locating the hidden malware for an application, but not provide enough detail to the Oracle for a correct detection response. They would therefore be given a “Keep looking” response, potentially causing them to abandon further investigation of the intended hidden malware.

To address this, the potential Oracle responses were updated to the following:

- “Your report has sufficient detail, and it describes the malice deliberately implanted in the app.”
- “Your report has sufficient detail, but it describes an issue that is not the malice deliberately implanted in the app.”
- “You must increase the level of detail in your report before the Oracle will evaluate it.”

These updates allowed the Oracle to ask for further information from the analysts without inadvertently informing them that they were close to finding the actual malware. It is important

to note that requests from the Oracle for further details were made regardless of whether the analyst team was describing the intended malware or not. Otherwise, the analysts could use these requests as indicators of where to focus their analysis.

## 5.2 Experiment Frequency and Duration

When experimentally evaluating a program, the frequency and duration of the experiments is an important topic to consider. One key factor in determining the correct duration and frequency is the workload that the experiments place on the performers. During the experiments, the performers will focus a significant portion of their time on analyzing adversarial challenges. This can come at the cost of pausing their research and development efforts. As a program advances, the challenges will also become more difficult to analyze successfully, and require greater analysis time per challenge. Therefore, determining the proper balance between evaluating the performers, and allowing them ample time to advance their research, is paramount.

The frequency of the experiments in each engagement also has a direct impact on the utility of the data produced by them. If the experiments are held in quick succession, then the performers will have little time to update their tools based on any lessons learned from the preceding experiments. Their performance in the subsequent experiments might not provide measurements that significantly differ from what was previously measured.

During APAC, the initial experiment phase consisted of a four-week off-site experiment followed by a two-week off-site experiment three weeks later. The experiment phase concluded with a two-day on-site experiment the following month. This experiment duration and frequency placed a heavy time burden on the participants. Subsequent engagements reduced the experiments to one off-site experiment and one on-site experiment, both with shorter durations. The reduced experiment burden continued to provide valuable performance results while also allowing the R&D teams more time to improve their tools.

## 5.3 Application Acceptance Criteria

A set of Application Acceptance Criteria was developed for the APAC program with the purpose of avoiding potential technical issues with the challenges used in the experiments. This criterion also ensured that all parties to the experiments had the same expectations with regard to the technical aspects of the challenges, and helped to avoid confusion during the experiments.

There were multiple technical issues that were encountered during the initial APAC experiments that prompted the creation of these criteria. For example, there are numerous source code build systems that can be used in Android application development. Some of the R&D tools that were being developed for APAC standardized on one of these build systems, and were not compatible with the other systems. Therefore, when these teams encountered an application that used another build system, they were unable to analyze the application. By specifying which build systems were allowed in the Application Acceptance Criteria, this issue was resolved in

subsequent experiments. For further examples of the types of issues that were resolved using this approach, the list of acceptance criteria that was used for the APAC engagements can be found in Appendix A and B.

#### 5.4 Unintended Malware Scoring

During the experiments, the R&D teams reported numerous instances of malicious functionality in the applications that were not the intended malware. Following the conclusion of each experiment (starting with Engagement 3) an analysis of the unintended malware detections was undertaken. The goal of this analysis was to determine if the detection accuracy scores for the teams should be adjusted based on any unintended malware they reported. To accomplish this, all Oracle emails and result documents from each experiment were reviewed and a list of all reported malicious functionality for each application was created. This list was then summarized by combining issues that were reported by multiple teams.

Once completed, this unintended malware list was sent to DARPA in order to obtain guidance on which issues might be important enough to alter the detection accuracy scores. Based on this process, the detection accuracy scores were adjusted for only a few applications. These changes did not affect the experiment results significantly. These adjustments to the scoring only modified some team's accuracy scores within a 10% range.

#### 5.5 One-on-one tool demonstrations

One key lesson learned from Engagement 1 was the need to measure how relevant each tool was when analyzing an application. In previous experiments the R&D teams were asked to self-report if their tool was relevant in the result documents for each application they analyzed. This was a good first step in determining how much of the detection accuracy for each team was the result of the tool, and how much was the result of manual code analysis.

In Engagement 2 One-on-one tool demonstrations were introduced during which each R&D team sat down with Five Directions and the AC Teams and analyzed a set of challenge applications together. The goal of these demonstrations was twofold: to get a better indication of how effective each R&D tool was by directly observing it in use, and to impart a better understanding of how to use each tool to the observers. These demonstrations also provided clarity about the portions of the tools that were most effective for each team.

## APPENDIX A – Original Application Acceptance Criteria

### Overview

All applications delivered for APAC Engagements 1 through 3 were checked against the list of requirements below. AC teams were informed of any apps that did not pass these criteria and were asked to re-submit those apps once they were fixed.

### Source

1. No non-java code in app
  - a. Interpreters written in Java are allowed
  - b. Resource files can contain malware
2. No code obfuscation - code needs to be readable
3. App LOC should be no less than 1k
4. No use of library projects
5. Include source for all non-Android .jar files as part of the project

### Compilation

1. Must compile with Eclipse
  - a. Provide import and build instructions specific to each app
  - b. Standardize on Eclipse 4.2.2, ADT 21.1
2. Must compile on the command line using ant
  - a. Provide build instructions for each app
  - b. Use JDK 6 (update 43)
  - c. Standardize on Ant 1.8.2
3. No use of other build systems
4. Must build with API level 15 - Android 4.0.3 (Google APIs level 15 supported as well)

### Run-time

1. App must load and run in the emulator
  - a. Standardize on the Galaxy Nexus emulator device with API level 15 as the target
  - b. Apps that use features not available on the emulator are allowed
    - i. Limit the number of apps that use non-emulator supported features to less than half of all delivered apps
2. Malware must be verifiable
  - a. Include instructions for triggering the malware

### Documentation

1. As with Engagement 1, AC Team app documents must be filled out for each app

## APPENDIX B – Updated Application Acceptance Criteria

### Overview:

All applications delivered for APAC Engagements 4 through 6 were checked against the list of requirements below. AC teams were informed of any apps that did not pass these criteria and were asked to re-submit those apps once they were fixed.

### Source:

6. No non-java code in app
  - a. Interpreters written in Java are allowed
  - b. Resource files can contain malware
7. No code obfuscation - code needs to be readable
8. App LOC should be no less than 1k
9. No use of library projects
10. Include the source code for any third party library as part of the application's source code

### Android API Support

5. Apps must build with Android 4.4.2 KitKat - API level 19
6. Google APIs level 19 may be used
7. Google Play Services revision 17 may be used

### Compilation:

1. Must compile with Eclipse
  - a. Provide import and build instructions specific to each app
  - b. Standardize on Eclipse 4.3.1, ADT 22.6.2
2. Must compile on the command line using ant
  - a. Provide build instructions for each app
  - b. Use JDK 6
  - c. Standardize on Ant 1.9.2
3. No use of other build systems

### Run-time:

3. App must load and run in the emulator or on actual hardware
  - a. Standardize on the Nexus 4 emulator device with Android 4.4.2 as the target platform
4. Malware must be verifiable
  - a. Include instructions for triggering the malware

### Documentation:

2. AC Team app documents must be filled out for each app

## APPENDIX C – Data Collection Forms

### AC Team Application Data Form

Experiment Application Name:

Archive Filename (containing source and .apk):

SHA-256 Hash of Archive:

Application Overt Purpose (*App store style description*):

-----

--

*(The E Team will provide the data above this line to the R&D and Control Teams)*

Application contains malicious functionality: ☐ YES ☐ NO

Malicious functionality location: (*filenames, class names, function names, line numbers*):

Malicious functionality triggers (summary):

Malicious effect when triggered (summary):

## R&D and Control Team Data Collection Form

Experiment Application Name:

### Effort:

Experiment Configuration Effort (person hours):

Analysis Effort (person hours):

### Result:

Application contains malicious functionality: ☐ YES ☐ NO

*(note: absence of evidence of malicious functionality = "no")*

Malicious functionality locations: *(filenames, class names, function names, line numbers)*:

Malicious functionality triggers (summary):

Malicious effects when triggered (summary):

Was your tool relevant in finding the threats, or lack thereof, in this application?

☐ YES ☐ NO

If yes, explain how:

If not, can the tool be modified to help find the threats or help show that the app is benign?

## LIST OF SYMBOLS, ABBREVIATIONS, AND ACRONYMS

AC Team	Adversarial Challenge Team, responsible for creating malicious applications
APAC	Automated Program Analysis for Cybersecurity
BAE	BAE Systems
BBN	Raytheon BBN Technologies
DARPA	Defense Advanced Research Projects Agency
GB	Gigabyte
ISU	Iowa State University
MIT	Massachusetts Institute of Technology
PI Meeting	Principle Investigator meeting
R&D Team	Research and Development Team, responsible for creating tools to detect malicious functionality
UCSB	University of California, Santa Barbara
UW	University of Washington